We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Extended Kalman Filter Based Fuzzy Adaptive Filter

Wai Kit Wong and Heng Siong Lim
Faculty of Engineering and Technology, Multimedia University, Jln Ayer Keroh Lama, 75450 Melaka, Malaysia

1. Introduction

In classical logic, every statement is either true or false, i.e., it has a truth value of 1 or 0. Classical sets impose rigid membership requirements. Fuzzy logic, which is the principle of imprecise knowledge, was introduced by Lofti A. Zadeh in 1965 (Zadeh, L. A., 1965). It is an extension of classical logic dealing with the partial truth concept. Every statement in fuzzy logic is a matter of degree and exact reasoning is viewed as a limiting case of approximate reasoning. In fuzzy logic, classical/Boolean truth value is replaced with degree of truth. Degree of truth denotes the extent to which a proposition is true. In fuzzy logic, the degree of truth of a proposition may be any real number between 0 and 1, inclusive. This fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition.

Fuzzy logic allows for set membership values between and including 0 and 1, shades of grey as well as black and white, and in its linguistic form, imprecise concepts like "slightly", "quite" and "very". Specifically it allows partial membership in a set. It is related to fuzzy sets and possibility theory. Fuzzy sets are an extension of classical set theory and are used in fuzzy logic (Zadeh, L. A., 1975). In classical set theory, the membership of elements in relation to a set is assessed in a crisp condition: either belongs to or not. In contrast, fuzzy set theory allows the gradual assessment of the membership of elements in relation to a set, with the aid of a membership function μ. A membership function may act as an indicator function, mapping all elements of fuzzy sets to real numbered value in the interval 0 and 1: $\mu \rightarrow [0,1]$. In general, there are 6 types of membership functions as depicted in Fig. 1.

2. Fuzzy logic system

A fuzzy logic system is composed of four principal components: a fuzzifier, a fuzzy rule base, a fuzzy inference engine and a defuzzifier (Mendel, J. M., 1995). It is an information processing system. Fig. 2 depicts a fuzzy logic system that is widely used in fuzzy logic controllers and signal processing applications.

The crisp inputs s, are first converted into fuzzy quantities u. This process is known as fuzzification, where a fuzzifier transforms crisp input values into linguistic values. Input values are translated into linguistic concepts, which are represented by fuzzy sets. Rules may be provided by experts or can be extracted from numerical data. In either case, engineering rules are expressed as a collection of IF-THEN statements. The inference engine

Source: Kalman Filter: Recent Advances and Applications, Book edited by: Victor M. Moreno and Alberto Pigazo,
ISBN 978-953-307-000-1, pp. 584, April 2009, I-Tech, Vienna, Austria

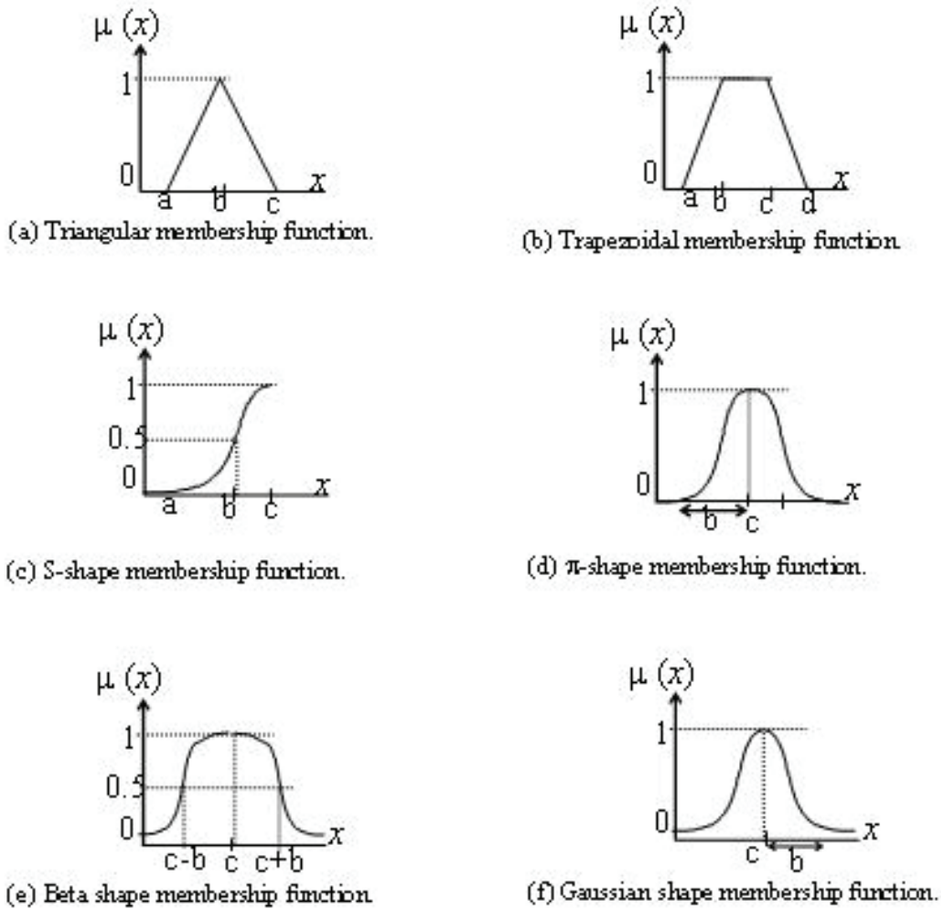

Fig. 1. Types of membership functions

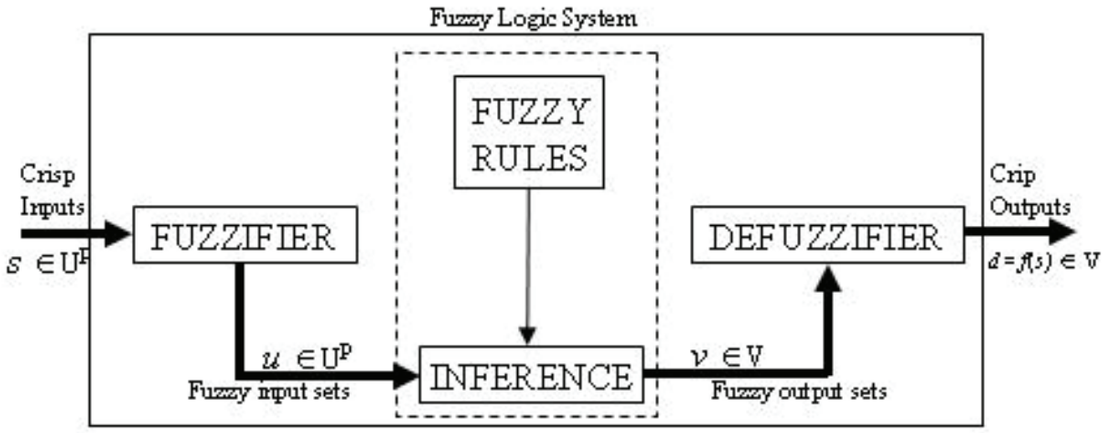

Fig. 2. Fuzzy logic system (Mendel, J.M., 1995)

of the fuzzy logic system maps fuzzy sets u into fuzzy sets v. It handles the way in which rules are combined. The defuzzifier maps output sets into crisp numbers. This mapping can be expressed quantitatively as $d = f(s)$. We shall consider an example of a simple

temperature regulator that uses a fan. Fig. 3 shows the fuzzy sets and the membership functions for the input temperature.

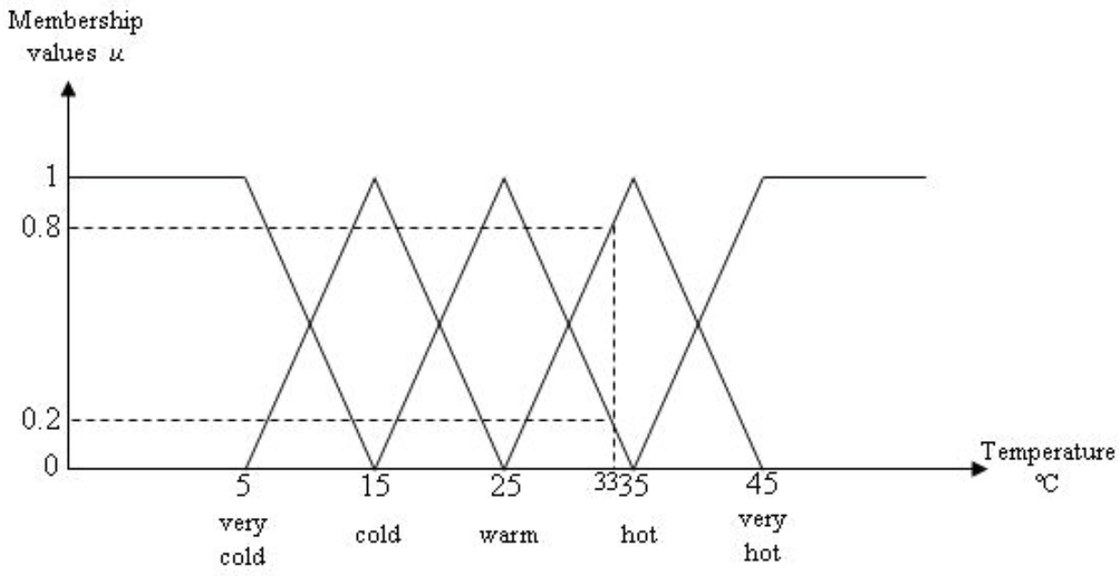

Fig. 3. Fuzzy sets of a simple temperature regulator that use a fan

A membership function μ_x, for $x \in X$, quantifies the grade of membership of the elements x to the fundamental set X. An element mapping to the value 0 means that the member is not included in the given set, 1 describes a fully included member. Values strictly between 0 and 1 characterize the fuzzy members. Membership functions are applied to the measurement and the degree of truth in each determined premise. According to Fig.3, if the input temperature s is 33ºC, after fuzzification, the membership value that s belongs to warm and hot temperature are $\mu_{warm} = 0.2$ and $\mu_{hot} = 0.8$ respectively.

The fuzzy rules may be provided by a human expert, or can be extracted from numerical input-output data pairs. In either case, engineering rules are expressed as a collection of IF-THEN statements, i.e.,

"IF temperature is very hot THEN speed maximum fan."
"IF temperature is hot THEN speed medium high fan."
"IF temperature is warm THEN maintain medium fan."
"IF temperature is cold THEN turn down medium low fan."
"IF temperature is very cold THEN stop fan."

These rules reveal that we will need an understanding of (Mendel, J.M., 1995):

1. Linguistic variables versus numerical values of the variable, e.g., very hot versus 45ºC.
2. Quantifying linguistic variables. E.g., u may have a finite number of linguistic terms associated with it, ranging from very hot to very cold, which is fuzzifying using fuzzy membership functions. There is no unique membership function in a situation and it is primarily subjective in nature. But this does not mean that membership function can be assigned arbitrarily, it is rather on the basis of application-specific criteria. Some of the commonly used membership functions are shown in Fig. 1 (Chen, S., 1990).
3. Implications, which is the relationship between two statements where the truth of one suggests the truth of the other, e.g., "IF temperature is warm THEN maintain medium

fan." Here, the truth of temperature is warm suggests the fan to maintain in medium speed.

4. Logical connections for linguistic variables, e.g., "and", "or", etc. "IF temperature is very hot and humidity is high, THEN speed medium high fan." Humidity is another fuzzy set. The fan speeds to medium high only when the two conditions are fulfilled.

Inference is the act or process of drawing a conclusion based solely on the fuzzy rules, e.g., u has $\mu_{warm} = 0.2$ and $\mu_{hot} = 0.8$, inference machine will draw a conclusion that the temperature is hot since $\mu_{hot} > \mu_{warm}$. Defuzzifier converts the fuzzy value into a "crisp" value. The defuzzifier maps output sets v into crisp numbers d. In the example mentioned above, v is speed medium high fan, when map into crisp number, d is the supply voltage value to the fan (e.g., 80 volts).

In summary, once the fuzzy rules have been established, the fuzzy logic system could map crisp inputs s into crisp outputs d. The mapping can be expressed quantitatively as $d = f(s)$. The fuzzifier maps crisp numbers s into fuzzy sets u in order to activate fuzzy rules that are in terms of linguistic variables, which have fuzzy sets associated with them. The inference machine maps fuzzy sets u into fuzzy sets v. It deals with the process in which rules are combined. The defuzzifier maps output sets v into crisp numbers d. In the sub-section below, we will discuss a well-known fuzzy logic system, which is the Takagi-Sugeno Kang (TSK) fuzzy logic system.

2.1 Takagi Sugeno Kang (TSK) fuzzy logic system

The Takagi-Sugeno Kang (TSK) fuzzy model is a universal approximator of the continuous real functions that are defined in a closed and bounded subset of n-dimensional real number $\Re^n$. This strong property of the TSK model finds several applications in modeling dynamical systems (Mastorakis, N.E., 2004). A TSK fuzzy logic system is described by fuzzy IF-THEN rules which represent input/output relations of a system. The most widely used TSK fuzzy logic system is the first-order TSK fuzzy logic system. It has a rule base of M rules, each having p antecedents, where the l-th rule is expressed as:

R^l: IF x_1 is F_1^l and x_2 is F_2^l and …and x_p is F_p^l

THEN $y^l = c_0^l + c_1^l x_1 + c_2^l x_2 + \ldots + c_p^l x_p$

in which l=1, 2,…, M; c_j^l is the consequent parameter, for j=0,1,…,p; x_j is the input to the fuzzy logic system; y^l is the output of the l-th IF-THEN rule; and F_j^l is the fuzzy sets, for j=0,1,…,p. The final output of the unnormalized first order TSK model is inferred as (Tanaka, K., 1998):

$$r = \sum_{l=1}^{M} f^l y^l \tag{1.1}$$

where f^l are rule firing strengths defined as:

$$f^l = \mathcal{T}_{j=1}^{p} \mu_{F_j^l}(x_j) \tag{1.2}$$

and $\mathcal{T}$ denotes a t-norm. t-norm is the short form of triangular norm. It is a kind of function used in multi-valued logic, especially in fuzzy logic. t-norm generalizes intersection in a lattice and AND in logic. The most often used t-norms are:

minimum *t*-norm : $\mathcal{T}_{min}(a,b) = \min\{a,b\}$

product *t*-norm : $\mathcal{T}_{prod}(a,b) = a.b$

drastic *t*-norm : $\mathcal{T}_{-1}(a,b) = \begin{cases} a & , \text{ if } b = 1 \\ b & , \text{ if } a = 1 \\ 0 & else \end{cases}$

When Gaussian membership functions

$$\mu_{F_j^l}(x_j) = \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^l}{\sigma_j^l}\right)^2\right] \tag{1.3}$$

and product *t*-norm are used, (1.1) can be expressed as:

$$r = \sum_{l=1}^{M} y^l \prod_{j=1}^{p} \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^l}{\sigma_j^l}\right)^2\right] \tag{1.4}$$

where m_j^l and σ_j^l are the centre and width of the *l*-th fuzzy set F_j^l, respectively.

2.2 Application of fuzzy logic system

Fuzzy logic systems deal with reasoning with fuzzy sets, fuzzy rules, and estimated sampled functions from linguistic input to linguistic output. Fuzzy logic systems are successful in the field of control especially the feed back control of some physical and chemical processes like electric current, temperature, motion of machines and flow of liquids or gas. Fuzzy logic principles are also be applied in fuzzy software engineering that incorporate fuzziness in data and programs and fuzzy database systems in the field of economics, management and medicines (Gupta, M.M., 1994[a], Gupta, M.M., 1994[b], Jang, J.S.R., 1993, Kaufmann, A., 1988). Recently, some automotive industry products and consumer electronics in the market have moved into fuzzy logic technology, and the outcome of the products has significant performance improvement (Al-Holou, N., 2002, Eichfeld, H., 1996).

Fuzzy logic systems are nonlinear systems and they are capable of inferring complex nonlinear relationships between input and output variables (Mendel, J.M., 1997). The non-linearity property is particularly important when the underlying physical mechanism to be modeled is inherently nonlinear. The system can 'learn' the non-linear mapping by being presented with a sequence of input signals and desired response pairs, which are used in conjunction with an optimization algorithm to determine the values of the system parameters. This is one of the most commonly used learning paradigms, called *supervised learning*. Even if the process to be modeled is non-stationary, the system can be updated to reflect the changing statistics of the process. Unlike conventional stochastic models used to model such processes, fuzzy logic systems do not make any assumptions regarding the structure of the process, nor do they invoke any kind of probabilistic distribution model, i.e., they belong to the general family of model free, data driven, non-parametric methods.

However, conventional fuzzy systems have limitation of low capabilities for learning and adaptation. An improvement done in (Gupta, M.M.,1991) combines conventional fuzzy

technology with neural network technology to form an innovative technological field, so called fuzzy neural networks (FNNs). Fuzzy mathematics gives an inference mechanism for approximate reasoning under cognitive uncertainty, while neural networks have the abilities of pattern recognition, optimization and decision making (Bhatti, S.S., 2002). A combination of these two technological innovations give birth to a new technology in which the explicit knowledge representation of fuzzy logic is improved by the learning power of simulated neural networks (Bhatti, S.S., 2002). Fuzzy basis function network (FBFN) is one of the FNNs that are used for information processing. It rediscovers some interesting advantages of a fuzzy system. One is the universal approximation capability, and the other is learning and adaptation, which have not been dealt with in fuzzy systems. Fuzzy adaptive equalizers are based on such technology. This technology is also capable of dealing with nonlinearity and uncertainty.

3. Equalizer

Consider, for example, a communication system which consists of a transmitter, communication channel, receiver, and equalizer connected together as shown in Fig. 4. The term equalizer in communication system is commonly refer to a device that place after the receiver designed to equalize the channel characteristics and extract out information send by transmitter side from noise and distortion.

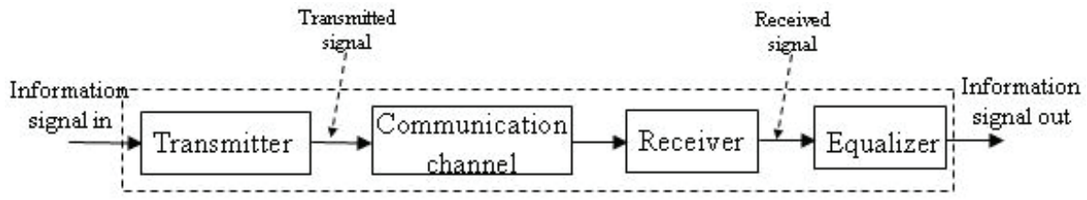

Fig. 4. Block diagram of a communication system

The main problem that has to be considered in communication system involves the fading and intersymbol interference (ISI), which is generated by multipath propagation effects and its resulting delay spread. Sometimes, there are obstacles and reflectors in the communication channel. The transmitted signal arrives at the receiver from various directions over a multiplicity of paths. Such a phenomenon is called multipath. It is an unpredictable set of reflections from direct waves and each with its own degree of attenuation and delay.
In communication channel, multiple reflections of the transmitted signal may arrive at the receiver at different times, this can result in intersymbol interference (or bits "crashing" into one another) which the receiver cannot sort out. This time dispersion of the channel is called multipath delay spread which is an important parameter to degrade the performance of communication systems.Besides signal distortion due to multipath propagation, noise is the most crucial factor that degrades the performance of communication systems. Since communication systems have such a lot of undesired distortions, equalizers are designed to compensate these distortions.

3.1 Optimal equalizer (MAP equalizer)

Maximum a-posteriori (MAP) equalizer is the optimal equalizer based on maximum a posteriori probability estimation. MAP equalization requires the knowledge of the

conditional probability density function (PDF) of the received signal given the transmitted signal pattern. Assume that we want to estimate an unobservable population θ on the basis of observations y. Let f be the sampling distribution of y, so that $f(y \mid \theta)$ is a conditional PDF, summarizing our knowledge provided by the data y conditioned on knowing θ. Then the function:

$$\theta \mapsto f(y \mid \theta) \tag{1.5}$$

is known as the likelihood function and the estimate (Kay, S.M., 1993):

$$\hat{\theta}_{ML}(y) = \arg \max_{\theta} f(y \mid \theta) \tag{1.6}$$

as the maximum likelihood estimate of θ. Now, assume that a prior distribution g over θ exists. Applying Bayesian statistics, we may treat θ as a random variable with posterior distribution as:

$$\theta \mapsto \frac{f(y \mid \theta)g(\theta)}{\int_{\theta} f(y \mid \theta)g(\theta)d\theta} \tag{1.7}$$

This is an application of Bayes' theorem. The method of maximum a-posteriori estimation is then estimates θ as the mode of the posterior distribution with:

$$\hat{\theta}_{MAP}(y) = \arg \max_{\theta} \frac{f(y \mid \theta)g(\theta)}{\int_{\theta} f(y \mid \theta)g(\theta)d\theta} = \arg \max_{\theta} f(y \mid \theta)g(\theta) \tag{1.8}$$

MAP estimate of θ coincides with maximum likelihood estimate when the prior distribution function g is uniform. The geometric formulation of MAP equalizer is given below.

3.2 Geometric formulation for MAP equalizer

The geometric formulation of the MAP equalizer is shown in (Chen, S., 1990) and (Chen, S., 1991). We are using the same notation as in those studies and we define:

$$P_{\eta,\tau}(1) = \{\hat{\mathrm{x}}(k) \in \Re^{\eta} \mid s(k-\tau) = 1\} \tag{1.9}$$

and

$$P_{\eta,\tau}(-1) = \{\hat{\mathrm{x}}(k) \in \Re^{\eta} \mid s(k-\tau) = -1\} \tag{1.10}$$

where η is the order, τ is the lag of the equalizer, $\Re$ is any real number,

$$\hat{\mathrm{x}}(k) = [\hat{x}(k) \quad \hat{x}(k-1) \quad \ldots \quad \hat{x}(k-\eta+1)]^T \tag{1.11}$$

$\hat{x}(k)$ is the noise-free output of the rake receiver, and $P_{\eta,\tau}(1)$ and $P_{\eta,\tau}(-1)$ are two sets of possible noise free output vectors $\hat{\mathrm{x}}(k)$ that can be produced from input sequences containing $s(k\text{-}\tau) = 1$ and $s(k\text{-}\tau) = -1$, respectively. Let:

$$\mathrm{x}(k) = [x(k) \quad x(k-1) \quad \ldots \quad x(k-\eta+1)]^T \tag{1.12}$$

be the observed output vector with noise, $p_1[\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)]$ and $p_{-1}[\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1)]$ be the conditional probability density functions of $\boldsymbol{x}(k)$ given $\hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)$ and $\hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1)$ respectively. It was shown in (Chen, S., 1990) and (Chen, S., 1991), which the MAP equalizer is defined by:

$$f_{opt}(\mathrm{x}(k)) = \mathrm{sgn}[p_1(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)) - p_{-1}(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1))] \tag{1.13}$$

This optimal equalizer achieves the minimum bit error rate for the given order η and τ, where sgn(q) = 1 (-1), if $q \geq 0$ (q<0). If the noise is Gaussian and with covariance matrix:

$$Q = E\{[(n(k) \quad \ldots \quad n(k-\eta+1)][n(k) \quad \ldots \quad n(k-\eta+1)]^T\} \tag{1.14}$$

Then from $x(k) = \hat{x}(k) + n(k)$, we have:

$$p_1(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(1)) - p_{-1}(\mathrm{x}(k) \mid \hat{\mathrm{x}}(k) \in P_{\eta,\tau}(-1))$$

$$= \sum \exp\left[-\frac{1}{2}(\mathrm{x}(k) - \hat{\mathrm{x}}_+)^T Q^{-1}(\mathrm{x}(k) - \hat{\mathrm{x}}_+)\right] - \sum \exp\left[-\frac{1}{2}(\mathrm{x}(k) - \hat{\mathrm{x}}_-)^T Q^{-1}(\mathrm{x}(k) - \hat{\mathrm{x}}_-)\right] \tag{1.15}$$

where the first summation is over all the positive noise free points $\hat{\mathrm{x}}_+ \in P_{\eta,\tau}(1)$ whereas the second summation is over all the negative noise free points $\hat{\mathrm{x}}_- \in P_{\eta,\tau}(-1)$.

In practice, it is difficult to know or predict the PDF of the transmitted or received signals, and MAP equalizer is too complex for practical use. Therefore, attention has been given to the design of sub-optimum equalizers such as adaptive equalizers that are practical and have near optimal performance.

4. Adaptive equalizer

Adaptive equalizers are sub-optimum equalizers that rely on a recursive algorithm to perform satisfactory information extraction in a communication system in which the statistical characteristics about the input-output signal are not known. The algorithm will start from some predetermined set of initial conditions, representing whatever the system knows about the communication channel. In a stationary communication channel, after successive iterations of the algorithm, the equalizer will converges to the optimum solution in some statistical sense. In a non-stationary communication channel, the algorithm will also have the ability to track time variations in the statistic of the input data (Haykin, S. 2002). Some adaptive equalizers examples are shown below:

4.1 Transversal equalizer

Digital equalizer is well known and was already described in 1940 by H. J. Kallman in (Kallman, H.J., 1940). The invention relates to a simple and linear transversal equalizer for processing an analog signal with a number of stages to which level control devices are connected by uniformly spaced taps in a non-dispersive delay line. The transversal equalizer is typically implemented using digital circuitry, charged-coupled devices, or surface-acoustic wave devices. Due to its versatility and ease of implementation, the transversal equalizer has emerged as an essential signal processing structure in a wide variety of applications.

In signal processing, transversal equalizer is also known as tapped-delay line equalizer or finite–duration impulse response (FIR) equalizer (Proakis, J.G.,1995). The structure of an adaptive transversal equalizer is depicted in Fig. 5:

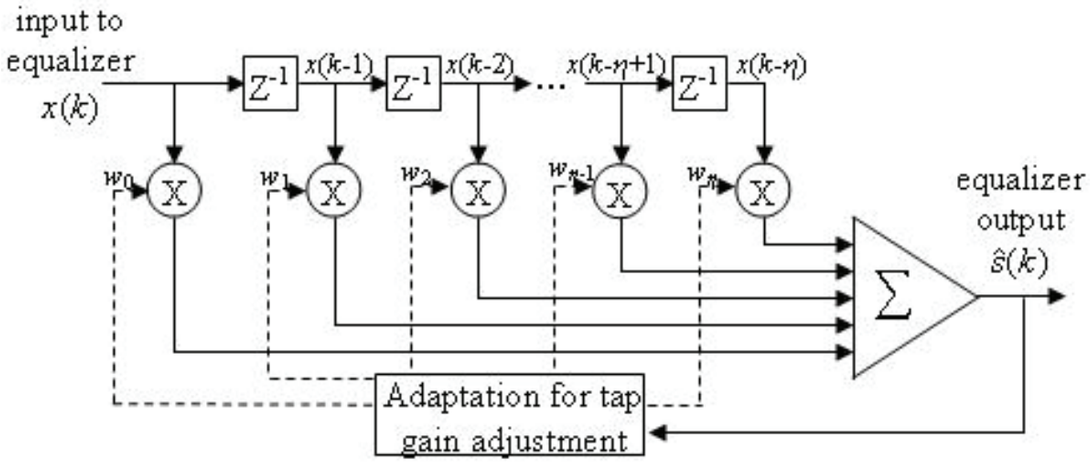

Fig. 5. Structure of transversal equalizer

The output of the transversal equalizer is given by:

$$\hat{s}(k) = \sum_{l=0}^{\eta} w_l x(k-l) \,. \tag{1.16}$$

where w_l is the l-th tap weight and η is the order of the equalizer. The weights of the equalizer can be optimized by minimizing some criterion functions. Two popular choices for the adaptation algorithm are the RLS and LMS algorithm.

4.2 Decision feedback equalizer

Another sub-optimum adaptive equalizer is the decision feedback equalizer (DFE). DFE utilizes two transversal equalizers: a feedforward transversal equalizer and a feedback transversal equalizer (Sailer, T.M., 2001). The DFE uses previous symbol estimates for interference cancellation of ISI corrupted data transmission. When the symbol estimates are correct, they have the advantage of not being corrupted by additive channel noise, giving the DFE a performance advantage over other linear structures. However, if the symbol estimates are incorrect, there is the danger of error propagation leading to catastrophic performance. The structure of DFE is shown in Fig. 6:

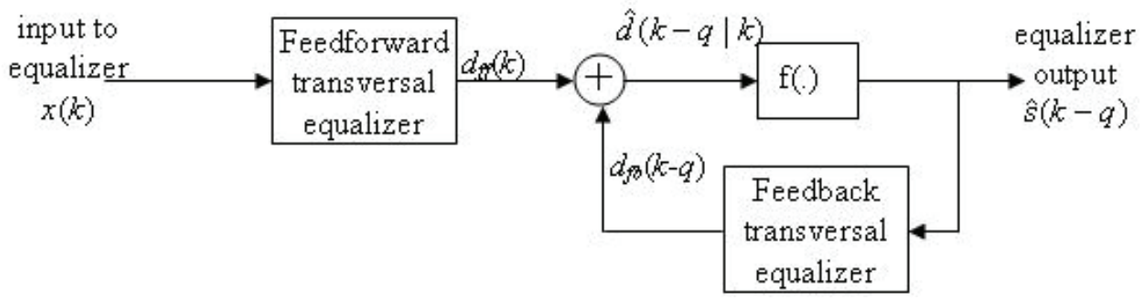

Fig. 6. Structure of decision feedback equalizer

where f(.) is a decision device and $x(k)$ is the input to the feedforward transversal equalizer. From the output of the feedforward transversal equalizer, $d_{ff}(k)$, the interference from previously detected symbols are removed via the output of the feedback transversal equalizer, $d_{fb}(k\text{-}q)$. The difference between these two transversal equalizer outputs constitutes an estimate of the transmitted symbol, $\hat{d}(k-q\,|\,k)$. This estimate is sometimes called *soft estimate*, since it is not yet quantized. The decision device quantizes the soft estimate and the resulting *hard estimate*, $\hat{s}(k-q)$ becomes the input of the feedback transversal equalizer. The constant q is known as the decision delay or the smoothing lag. It specifies how many future measurements are being processed before a decision is made on the present symbol.

4.3 Volterra series expansion equalizer

Since the optimal decision boundary is normally nonlinear for communication channel equalization problem, linear equalization methods are no longer adequate for the task. In this case, nonlinear equalizers that have the ability to perform nonlinear input-output mapping can be applied to minimize the error probability. In this and the following two subsections, we discuss three nonlinear channel equalization methods, namely the Volterra series expansion, the radial basis function and the multilayer perceptrons. The Volterra series expansion equalizer is a nonlinear equalizer based on the Volterra series functional representation from mathematics (Kong, X., 2004).

Let $x[n]$ and $y[n]$ represent the input and output signals, respectively, of a discrete time and causal nonlinear system. The Volterra series expansion for $y[n]$ using $x[n]$ is given by (Mathews, V.J., 1991):

$$y[n] = h_0 + \sum_{m_1=0}^{\infty} h_1[m_1]x[n-m_1] + \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty} h_2[m_1,m_2]x[n-m_1]x[n-m_2] + \ldots$$

$$+ \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}\ldots\sum_{m_p=0}^{\infty} h_p[m_1,m_2,\ldots,m_p]x[n-m_1]x[n-m_2]\ldots x[n-m_p] + \ldots \quad (1.17)$$

where $h_p[m_1,m_2,\ldots,m_p]$ is known as the p-th order Volterra kernel of the system. $h_p[m_1,m_2,\ldots,m_p]$ can be optimized by minimizing some criterion functions. Among the most commonly used algorithm are the recursive least squares (RLS) and least mean squares (LMS) algorithms. Without any loss of generality, it is assumed that the Volterra kernels are symmetric, in which $h_p[m_1,m_2,\ldots,m_p]$ is left unchanged for any of the possible p! permutations of the indices $m_1,m_2,\ldots,m_p$ (Mathews, V.J., 1991).

Since an infinite series expansion like (1.17) is not useful in channel equalization, we may work with truncated Volterra series expansion (Mathews, V.J., 1991):

$$y[n] = \sum_{m_1=0}^{N-1} h_1[m_1]x[n-m_1] + \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2[m_1,m_2]x[n-m_1]x[n-m_2] + \ldots$$

$$+ \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1}\ldots\sum_{m_p=0}^{N-1} h_p[m_1,m_2,\ldots,m_p]x[n-m_1]x[n-m_2]\ldots x[n-m_p] \quad (1.18)$$

where $h_0 = 0$ (without loss of generality (Mathews, V.J., 1991)). Note that there are $O(N^p)$ coefficients in this polynomial expansion. One big disadvantage for the model as in (1.18) is that the complexity of implementing equalizers using this model can be very large even for moderately large values of N and P. Consequently, most Volterra series expansions system involve low order model.

4.4 Radial basis function equalizer

A radial basis function (RBF) equalizer is a neural network equalizer whose outputs are a linear combination of the hidden layer functions (Mulgrew, B., 1996). It is trained to perform a mapping from an m-dimensional input space to an n-dimensional output space. The structure of RBF equalizer is shown in Fig. 7 below.

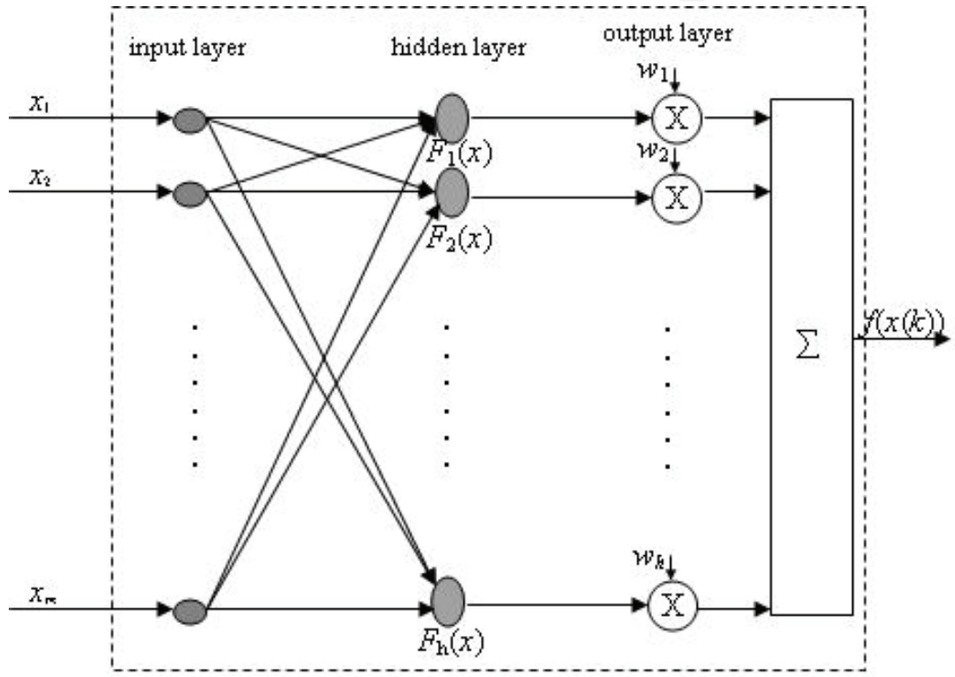

Fig. 7. Structure of radial basis function equalizer

Mathematically, the output of a RBF equalizer is,

$$f(x(k)) = \sum_{l=1}^{h} w_l F_l(\boldsymbol{x}) \tag{1.19}$$

where the basis function F is a sigmoidal function

$$F(\boldsymbol{x}) = \exp\left(-\frac{1}{\sigma_l^2} \left\| \boldsymbol{x} - \mu_l \right\|^2 \right) \tag{1.20}$$

$\boldsymbol{x}$ is the input vector of the equalizer [x_1 x_2 … x_m], h indicates the total number of hidden neurons, μ_l and σ_l refer to the center and width of the l-th hidden neuron. $||.||$ is the Euclidean norm. The coefficient w_l is the weight of the l-th hidden neuron to the output neuron.

4.5 Multilayer perceptrons equalizer

Another approach for nonlinear channel equalization using neural network is the multilayer perceptrons (MLP) equalizer. A typical MLP equalizer consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the MLP equalizer layer-by-layer. The signal-flow of such an equalizer is shown in Fig. 9. In between the input layer and the output layer are the hidden layers of the MLP equalizer. The MLP equalizer has L layers of synaptic connections and L+1 layers of neurons.

In (Seung, S., 2002), back propagation algorithm is used in multilayer perceptrons network. Assume there are no biases, the network is diagrammed as:

$$x^0 \xrightarrow{W^1} x^1 \xrightarrow{W^2} \cdots \xrightarrow{W^L} x^L \tag{1.21}$$

where $x^l \in \Re^{n_l}$ for all l = 0,1, …, L; $x^l = [x_1^l \; x_2^l \ldots x_h^l]$ and W^l is an $n_l \times n_{l-1}$ matrix for all l = 0,1, …, L. n_l is the number of hidden neurons in l-th layer. There are L + 1 layers of neurons. The input vector x^0 is transformed into the output vector x^L by evaluating the equation:

$$x_i^l = \sum_{j=1}^{n_l - 1} W_{ij}^l x_j^{l-1} \quad \text{for } l = 1 \text{ to } L. \tag{1.22}$$

The actual output of the equalizer is

$$f(x(k)) = \sum_{i=1}^{h} x_i^L \tag{1.23}$$

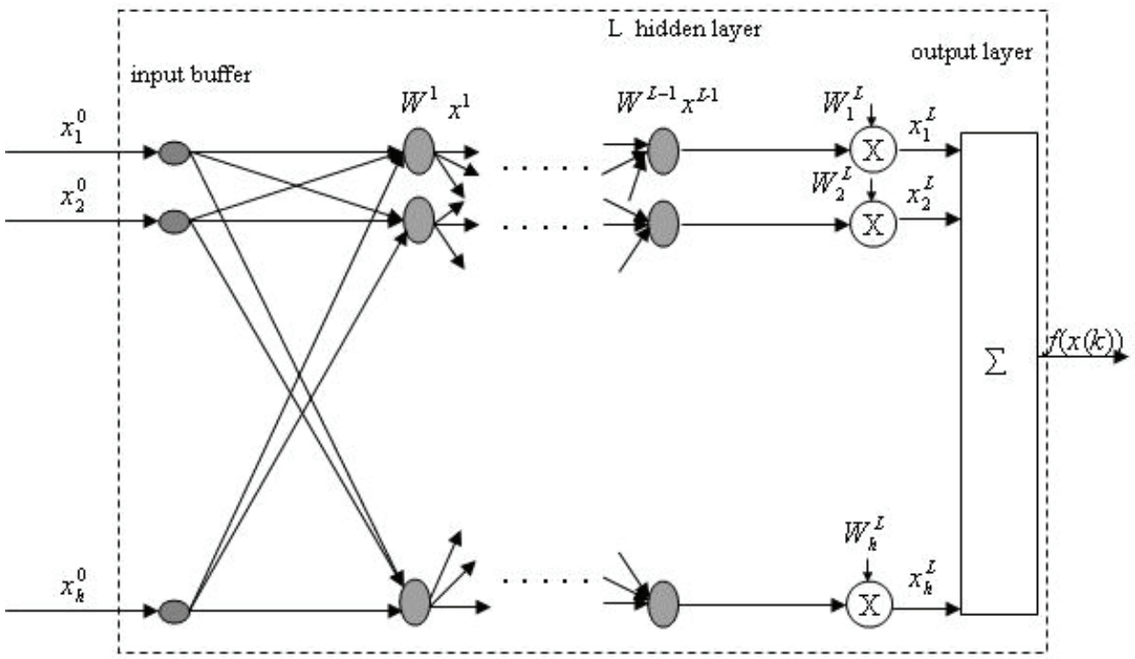

Fig. 9. Structure of multilayer perceptron equalizer

Two types of linear adaptive equalizer had been discussed, namely transversal equalizer and decision feedback equalizer; at the meantime, three types of nonlinear channel are also discussed, namely Volterra series expansion equalizer, radial basis function equalizer and multilayer perceptron equalizer. Normally, complex communication channels are nonlinear channel. Therefore most of the adaptive equalizers used today are nonlinear adaptive equalizer, because they not only dealing well with linear channel characteristic, but nonlinear channel as well. Since nonlinear channels include a very broad spectrum of nonlinear distortion, it is difficult to comment on which nonlinear adaptive equalizer is dominantly better than the others. So, it is good to try out new nonlinear adaptive equalizer for communication channel equalization. Fuzzy adaptive equalizer is such a new nonlinear equalizer.

5. Fuzzy adaptive equalizer

Fuzzy adaptive equalizers are adaptive equalizers that apply the concepts of fuzzy logic. Fuzzy adaptive equalizers are information processors that make use of both linguistic (in the form of fuzzy IF-THEN rules) and numerical information (in the form of input-output pairs). The main merits of using fuzzy adaptive equalizers are nonlinear and simple in design, which linguistic information from human experts can be directly incorporated into the equalizer. If no linguistic information is available, the fuzzy adaptive equalizers become well-defined nonlinear adaptive equalizers (similar to the polynomial, neural nets, or radial basis function adaptive equalizers). The adaptive algorithms adjust the parameters of the membership functions which characterize the fuzzy concepts in the IF-THEN rules, by minimizing some criterion function.

Fuzzy adaptive equalizer, as a fuzzy basis function expansions system, can be represented as two-layered feedforward network structure (Wang, L.X., 1992[b]). On the basis of this idea, the fuzzy adaptive equalizer can be trained to realize the desired input-output relationship using various learning algorithms such as least mean squares (LMS), recursive least squares (RLS) and extended Kalman filter (EKF) adaptation algorithms. Fig. 10 shows the schematic diagram of fuzzy adaptive equalizer. The inputs to the fuzzy adaptive equalizer [$x(k)$, $x(k-1)$,…, $x(k-n+1)$] are the receiver's outputs. The task of the fuzzy adaptive equalizer at the sampling instant k is to produce an estimate of the transmitted symbol $\hat{s}(k-d)$, using the information contained in [$x(k)$, $x(k-1)$,…, $x(k-n+1)$] where the integer n and d are the order and lag of the equalizer respectively.

A fuzzy adaptive equalizer is functionally equivalent to a fuzzy basis function network (FBFN) with the form given in (1.24) to (1.27). It can be shown that the input-output equations of a fuzzy adaptive equalizer with singleton fuzzifier, product inference and centroid defuzzifier can be expressed as:

$$f(\boldsymbol{x}(k)) = \frac{\sum_{l=1}^{M} \theta^{l} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)} = \sum_{l=1}^{M} \theta^{l} \phi^{l}(\mathrm{x}) \tag{1.24}$$

$$\text{where} \quad \phi^l(\mathrm{x}) = \frac{\prod_{i=1}^{n} \mu_{F_i^l}(x_i)}{\sum_{l=1}^{M}\left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right)} \tag{1.25}$$

$\phi^l(\mathrm{x})$ are called the fuzzy basis functions. Equation (1.24) gives an expression for fuzzy adaptive equalizer as shown in Fig.10. $\boldsymbol{x}$ =[x_1 x_2 …x_{n-1}]=[$x(k)$ $x(k\text{-}1)$ … $x(k\text{-}n\text{+}1)$] is the vector of inputs to the fuzzy adaptive equalizer. In particular, if a Gaussian radial basis function is chosen as the membership function, then

$$\phi^l(\mathrm{x}) = \frac{u_l(\mathrm{x})}{\sum_{l=1}^{M} u_l(\mathrm{x})} \tag{1.26}$$

$$\text{where} \quad u_l(\boldsymbol{x}) = \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right) = \prod_{i=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)\right) \tag{1.27}$$

$u_l(\boldsymbol{x})$ is the membership function after product inference, $\tilde{x}_i^l$ is the center of the i-th membership function and σ_i^l represents the width of the i-th membership function. In the next two sections, we will discuss two examples of fuzzy adaptive equalizers that were proposed in (Wang, L.X., 1993) for nonlinear channel equalization.

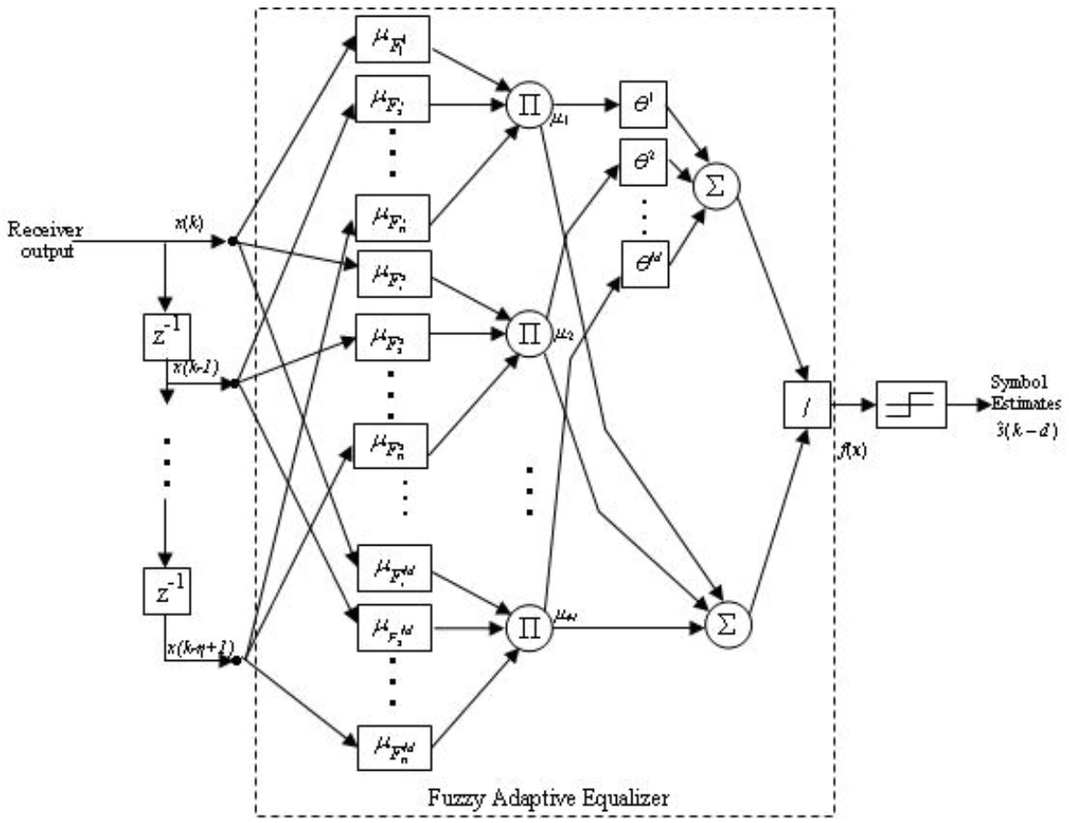

Fig. 10. Schematic diagram of fuzzy adaptive equalizer.

6. RLS based fuzzy adaptive equalizer

In (Wang, L.X., 1993), the RLS based fuzzy adaptive equalizer is used to solve the following problem. Consider a real-valued vector input sequence [$\boldsymbol{x}(k)$] and a real-valued scalar sequence [$d(k)$], where k=0,1,2,… is the time index, and $\boldsymbol{x}(k) \in U \equiv [C_1^-, C_1^+] \times [C_2^-, C_2^+] \times \ldots \times [C_n^-, C_n^+] \subset \Re^n$. U and $\Re$ are the input and output spaces of the equalizer respectively. At each time point k, the values of $\boldsymbol{x}(k)$ and $d(k)$ are given. The problem to be solved is to determine a fuzzy adaptive equalizer $f_k : U \subset \Re^n \to \Re$ such that:

$$J(k) = \sum_{i=0}^{k} \lambda^{k-i} [d(i) - f_k(\boldsymbol{x}(i))]^2 \tag{1.28}$$

is minimized, where $\lambda \in (0,1]$ is a forgetting factor.

6.1 Design procedure of the RLS based fuzzy adaptive equalizer

Step 1: m_i fuzzy sets are defined in each interval $\left[C_i^-, C_i^+\right]$ of the input space U, which are labeled as F_i^{ji} (i = 1, 2, …, n; ji = 1, 2, …, m_i; ji is a single index, i.e., $j1$ is an index which takes values from 1 to m_1 for i=1), in the following way: the m_i membership functions $\mu_{F_i^{ji}}$ cover the interval $\left[C_i^-, C_i^+\right]$ in the sense that for each $x_i \in \left[C_i^-, C_i^+\right]$ there exist at least one $\mu_{F_i^{ji}}(x_i) \neq 0$. These membership functions are fixed and will not change during the adaptation procedure.

Step 2: A set of $\prod_{i=1}^{n} m_i$ fuzzy IF-THEN rules is constructed in the following form:

$$R^{(j1,\ldots,jn)}: \text{IF } x_1 \text{ is } F_1^{j1} \text{ and } \ldots \text{ and } x_n \text{ is } F_n^{jn}, \text{ THEN } d \text{ is } G^{(j1,\ldots,jn)}. \tag{1.29}$$

where $\mathrm{x} = \left[x_1,\ldots,x_n\right]^T = \left[x(k),\ldots,x(k-n+1)\right]^T \in U$ is the equalizer input, $d \in \Re$ is the equalizer output, ji = 1, 2, …, m_i with i =1, 2, …, n, F_i^{ji}'s are the same labels of the fuzzy sets defined in Step 1, and the $G^{(j1,\ldots,jn)}$'s are labels of fuzzy sets defined in the output space which are determined in the following way: if there are linguistic rules from human experts in the form of (1.29), $G^{(j1,\ldots,jn)}$ is set to the corresponding linguistic terms of these rules; otherwise, $\mu_{G^{(j1,\ldots,jn)}}$ is set to an arbitrary membership function over the output space $\Re$. It is in this way that linguistic rules are incorporated into the fuzzy adaptive equalizer.

Step 3: The filter output f_k is calculated based on the $\prod_{i=1}^{n} m_i$ rules in step 2 as follows:

$$f_k(\mathrm{x}) = \frac{\sum_{j1=1}^{m_1} \ldots \sum_{jn=1}^{m_n} \theta^{(j1,\ldots,jn)} \left(\mu_{F_1^{j1}}(x_1) \ldots \mu_{F_n^{jn}}(x_n)\right)}{\sum_{j1=1}^{m_1} \ldots \sum_{jn=1}^{m_n} \left(\mu_{F_1^{j1}}(x_1) \ldots \mu_{F_n^{jn}}(x_n)\right)} \tag{1.30}$$

where $\mathrm{x} = [x_1,...,x_n]^T = [x(k),...,x(k-n+1)]^T \in U$, $\mu_{F_i^{ji}}$'s are membership functions defined in Step 1, and $\theta^{(j1,...,jn)} \in \Re$ is the point at which $\mu_{G^{(j1,...,jn)}}$ achieves its maximum value. Due to the way in which the $\mu_{F_i^{ji}}$'s are defined in Step 1, the denominator of (1.30) is nonzero for all the points of *U*; therefore the filter f_k of (1.30) is well defined. For a given input $\mathrm{x} \in U$, the equalizer output is determined as a weighted average of the $\prod_{i=1}^{n} m_i$ points $\theta^{(j1,...,jn)}$ in the output space at which the fuzzy sets $G^{(j1,...,jn)}$ of the THEN parts of the $\prod_{i=1}^{n} m_i$ rules have maximum membership values; and, the weight $\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)$ for $\theta^{(j1,...,jn)}$ is proportional to the membership values of which ***x*** satisfies the IF part of $R^{(j1,...,jn)}$.

6.2 Parameter adaptation of the RLS based fuzzy adaptive equalizer

In (1.30), the weights $\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)$ are fixed functions of ***x***; therefore the free design parameters of the fuzzy adaptive equalizer are the $\theta^{(j1,...,jn)}$'s which are collected as a $\prod_{i=1}^{n} m_i$ -dimensional vector (Wang, L.X., 1993):

$$\theta = [\theta^{(1,1,...,1)},...,\theta^{(m_1,1,...,1)},\theta^{(1,2,...,1)},...,\theta^{(m_1,2,1,...,1)},...,\theta^{(1,m_2,1,...,1)},...,\theta^{(m_1,m_2,1,...,1)},...,\theta^{(1,m_2,...m_n)},...,\theta^{(m_1,m_2,...,m_n)}]^T \quad (1.31)$$

The fuzzy basis function is defined by

$$p^{(j1,...,jn)}(\mathrm{x}) = \frac{\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)}{\sum_{j1=1}^{m_1}...\sum_{jn=1}^{m_n}\left(\mu_{F_1^{j1}}(x_1)...\mu_{F_n^{jn}}(x_n)\right)} \quad (1.32)$$

and they are collected as a $\prod_{i=1}^{n} m_i$ -dimensional vector $\boldsymbol{p}(\boldsymbol{x})$ in the same ordering as the $\boldsymbol{\theta}$ of (1.31), i.e.,

$$\mathrm{p}(\mathrm{x}) = (p^{(1,1,...,1)}(\mathrm{x}),..., p^{(m_1,1,...,1)}(\mathrm{x}), p^{(1,2,1,...,1)}(\mathrm{x}),..., p^{(m_1,2,1,...,1)}(\mathrm{x}),..., p^{(1,m_1,1,...,1)}(\mathrm{x})$$
$$,..., p^{(m_1,m_2,1,...,1)}(\mathrm{x}),..., p^{(1,m_2,...,m_n)}(\mathrm{x}),..., p^{(m_1,m_2,...,m_n)}(\mathrm{x}))^T \quad (1.33)$$

Based on (1.31) and (1.33), (1.30) can be rewritten as:

$$f_k(\boldsymbol{x}) = \mathrm{p}^T(\mathrm{x})\theta \quad (1.34)$$

The following RLS algorithm is used to update θ. The initial estimate of $\boldsymbol{\theta}$, $\boldsymbol{\theta}(0)$, is determined as in Step 2, and $P(0)=\sigma I$, where σ is a small positive constant, and I is the $\prod_{i=1}^{n} m_i$ -by-$\prod_{i=1}^{n} m_i$ identity matrix. At each time point k=1,2, ..., the following equation are computed:

$$\phi(k) = \mathrm{p}(\mathrm{x}(k)) \tag{1.35}$$

$$P(k) = \frac{1}{\lambda}\left[P(k-1) - P(k-1)\phi(k)[\lambda + \phi^T(k)P(k-1)\phi(k)]^{-1}\phi^T(k)P(k-1)\right] \tag{1.36}$$

$$K(k) = P(k-1)\phi(k)[\lambda + \phi^T(k)P(k-1)\phi(k)]^{-1} \tag{1.37}$$

$$\theta(k) = \theta(k-1) + K(k)(d(k) - \phi^T(k)\theta(k-1)) \tag{1.38}$$

where $\boldsymbol{x}(k)$ is the real-valued input vector and $d(k)$ is the real-valued desired output scalar sequence. $\mathrm{p}(\mathrm{x}(k))$ is defined in (1.33), and λ is the forgetting factor. Some comments on the RLS based fuzzy adaptive equalizer are given in the next sub-section.

6.3 Comments on RLS based fuzzy adaptive equalizer

The RLS algorithm, (1.36) – (1.38) are obtained by minimizing the recursive least squares criterion, $J(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - f_k(\mathrm{x}(i))]^2$, with f_k constrained to be the form of (1.34). Because f_k is linear in the parameter, the derivation of (1.36) – (1.38) is the same as that for the FIR linear adaptive filter (Cowan, C.F.N., 1985).

Equations (1.36) – (1.38) can be viewed as updating the $\prod_{i=1}^{n} m_i$ rules in the form of (1.29) by changing the centers $\theta^{(j1,\ldots,jn)}$ of the THEN parts of these rules in the direction that minimizing the criterion function $J(k)$ as in (1.28). Only these centers are allowed to change. The membership functions $\mu_{F_i^{ji}}$ of the IF parts of the rules are fixed at the very beginning and are not allowed to change. Hence, a good choice of the membership functions is very important to the success of the entire equalizer. However, in the next section, LMS based fuzzy adaptive equalizer will lighten this constraint by allowing $\mu_{F_i^{ji}}$'s also to change during the adaptation process.

It was proven in (Wang, L.X., 1992[a]) and (Wang, L.X., 1992[b]) that the equalizers (1.30) are universal approximators; i.e. for any real continuous function g on the compact set U, there exists a function in the form of (1.30) such that it can uniformly approximate g over U to an arbitrary accuracy. Therefore, this fuzzy adaptive equalizer is a powerful nonlinear adaptive equalizer in the sense that it has the capability of performing very difficult nonlinear equalization operation. This type of fuzzy adaptive equalizer performs two operations on the input vector $\boldsymbol{x}$: first, it performs a nonlinear transformation $\boldsymbol{p}(.)$ on $\boldsymbol{x}$; then the equalizer output is obtained as a linear combination of these transformed signals. The fuzzy adaptive equalizer is similar to the radial basis function (Chen, S., 1991), (Powell, M.J.D., 1987) and potential function (Meisel, W.S., 1969) approaches.

Linguistic information (in the form of fuzzy IF-THEN rules of (1.30)) and numerical information (in the form of desired input-output pairs $\boldsymbol{x}(k)$, $d(k)$) are combined into the equalizer in the following way: In Step 2-3, linguistic IF-THEN rules are directly incorporated into the equalizer (1.30) by constructing the initial equalizer based on the linguistic rules. During the parameter adaptation, numerical pairs ($\boldsymbol{x}(k)$, $d(k)$) are

incorporated into the equalizer by updating the equalizer parameters such that the equalizer output "matches" the input-output pairs in the sense of minimizing (1.28).
A nonlinear equalizer which is linear in the parameter is obtained by fixing the fuzzy membership functions on the input space U at the very beginning. So, the fast-convergent RLS algorithm could be used in the adaptation process, but has to include all the $\prod_{i=1}^{n} m_i$ possible rules in the equalizer because if a region of U is not covered by any rules and an input $\boldsymbol{x}$ to the equalizer happens to be in that region, then the equalizer response will be very poor. So, for problems of high dimension η and large m_i, the computations involved in this type of fuzzy adaptive equalizer are with higher complexity, because at each time point k, we need to perform the $\prod_{i=1}^{n} m_i$ -dimensional matrix-to vector multiplications of (1.36)-(1.38), and to evaluate the values of the $\prod_{i=1}^{n} m_i$ fuzzy basis functions of (1.35). Although these computations are highly parallelizable, we may not be able to use the equalizer in some practical situations where computing power is limited. Therefore, a LMS based fuzzy adaptive equalizer is developed in (Wang, L.X., 1993), which involves much less computation.

7. LMS based fuzzy adaptive equalizer

In (Wang, L.X., 1993), the LMS fuzzy adaptive equalizer solves the following problem. Consider the same input and output sequence, $x(k)$ and $d(k)$ as in RLS based fuzzy adaptive equalizer. At each time point k=1,2,…, a fuzzy adaptive equalizer is required to determine $f_k : U \to R$ such that

$$L = E[(d(k) - f_k(\mathrm{x}(k)))^2] \tag{1.39}$$

is minimized.

7.1 Design procedure of the LMS based fuzzy adaptive equalizer

Step 1: M fuzzy sets F_i^l in each interval $\left[C_i^-, C_i^+\right]$ of U are defined with the following Gaussian membership functions:

$$\mu_{F_i^l} = \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right] \tag{1.40}$$

where l = 1, 2, …, M, i = 1, 2, …, n, $x_i \in \left[C_i^-, C_i^+\right]$, and $\tilde{x}_i^l$ and σ_i^l are free parameters which will be updated in the LMS adaptation process.
Step 2: A set of M fuzzy IF-THEN rules is constructed in the following form:

$$R^l: \text{ IF } x_1 \text{ is } F_1^l \text{ and } \ldots \text{ and } x_n \text{ is } F_n^l, \text{ THEN } d \text{ is } G^l. \tag{1.41}$$

where $\mathrm{x} = [x_1, \ldots, x_n]^T = [x(k), \ldots, x(k-n+1)]^T \in U$, $d \in \Re$, F_i^l are defined in Step 1, $M \le \prod_{i=1}^{n} m_i$ and G^l's are fuzzy sets defined in $\Re$ which are determined as follows: if there

are linguistic rules in the form of (1.41), set F_i^l and G^l to be the labels of these linguistic rules; otherwise, choose μ_{G^l} and the parameters $\tilde{x}_i^l$ and σ_i^l arbitrarily. The parameters of membership functions $\mu_{F_i^l}$ and μ_{G^l} in these rules will change during the LMS adaptation process. Hence, the rules constructed in this step are initial rules of the fuzzy adaptive equalizer. Similar to RLS based fuzzy adaptive equalizer, linguistic rules are incorporated into the LMS based fuzzy adaptive equalizer by constructing the initial equalizer based on these rules.

Step 3: The filter f_k:$U \to R$ is constructed based on the *M* rules of Step 2 as follows:

$$f_k(\mathrm{x}) = \frac{\sum_{l=1}^{M} \theta^l \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i) \right)} \tag{1.42}$$

where $\mathrm{x} = [x_1, ..., x_n]^T = [x(k), ..., x(k-n+1)]^T \in U$, $\mu_{F_i^l}$'s are the Gaussian membership functions of (1.40), and $\theta^l \in \Re$ is any point at which μ_{G^l} achieves its maximum value. The equalizer in (1.42) is constructed in the same way as (1.30), and shares the same interpretation. Because the membership functions $\mu_{F_i^l}(x_i)$ is chosen to be Gaussian functions, which are nonzero for any $x_i \in [C_i^-, C_i^+]$, the denominator of (1.42) is nonzero for any $\mathrm{x} \in U$; therefore the equalizer in (1.42) is well defined. Because the θ^l as well as $\tilde{x}_i^l$ and σ_i^l are free parameters, the equalizer is nonlinear in the parameters.

7.2 Parameter adaptation of the LMS based fuzzy adaptive

The following LMS algorithm (Wang, L.X., 1993) is used to update the LMS based fuzzy adaptive equalizer parameters: θ^l, $\tilde{x}_i^l$ and σ_i^l. The initial $\theta^l(0)$, $\tilde{x}_i^l(0)$ and $\sigma_i^l(0)$ are determined in Step 2. At each time point *k* = 1, 2, …, do the following:

$$\theta^l(k) = \theta^l(k-1) + \alpha[d(k) - f_k] \frac{a^l(k-1)}{b(k-1)} \tag{1.43}$$

$$\tilde{x}_i^l(k) = \tilde{x}_i^l(k-1) + \alpha[d(k) - f_k] \frac{\theta^l(k-1) - f_k}{b(k-1)} a^l(k-1) \frac{x_i(k) - \tilde{x}_i^l(k-1)}{\left(\sigma_i^l(k-1)\right)^2} \tag{1.44}$$

$$\sigma_i^l(k) = \sigma_i^l(k-1) + \alpha[d(k) - f_k] \frac{\theta^l(k-1) - f_k}{b(k-1)} a^l(k-1) \frac{\left(x_i(k) - \tilde{x}_i^l(k-1)\right)^2}{\left(\sigma_i^l(k-1)\right)^3} \tag{1.45}$$

where $a^l(k-1) = \prod_{i=1}^{n} \exp\left[-\frac{1}{2} \left(\frac{x_i(k) - \tilde{x}_i^l(k-1)}{\sigma_i^l(k-1)} \right)^2 \right]$, $b(k-1) = \sum_{l=1}^{M} a^l(k-1)$, and

$f_k = \frac{\sum_{l=1}^{M} \theta^l a^l(k-1)}{b(k-1)}$. a is a small positive step size, l = 1, 2, …, M, and i =1, 2, …,n. Equations (1.43) - (1.45) are obtained by taking the gradient of L(ignore the expectation E) as in (1.39) with respect to the parameters and using the specific formula of (1.42) and (1.40). Some comments on the LMS based fuzzy adaptive equalizer are given in the next sub-section.

7.3 Comments on LMS based fuzzy adaptive equalizer

From Steps 2 and 3, the initial LMS based fuzzy adaptive equalizer is constructed based on linguistic rules from human experts and some arbitrary rules (in the sense that the parameters of membership functions $\mu_{F_i^l}$ and μ_{G^l} which characterize these rules are chosen arbitrarily). Both sets of rules are updated during the LMS adaptation process, changing the parameters in the direction of minimizing (1.39). Since minimizing (1.39) can be viewed as matching the input-output pairs $\{x(k),d(k)\}$, the LMS based fuzzy adaptive equalizer can be said to combine both linguistic and numerical information in its design.

LMS algorithm is a gradient algorithm. Therefore, a good choice of initial parameters is very important for its convergence. Since linguistic information is used to choose the initial parameters, the adaptation process will converge faster if the linguistic rules give good instructions for how the equalizer should perform, i.e., good description of the input-output pairs $\{x(k),d(k)\}$. Although LMS algorithm in general is slow to converge, LMS algorithm in fuzzy adaptive equalizer may converge fast, provided that there are sufficient linguistic rules.

A novel fuzzy adaptive equalizer that uses the extended Kalman filter (EKF) adaptation algorithm will be discussed in the next section. As compared to RLS and LMS based fuzzy adaptive equalizer, EKF based fuzzy adaptive equalizer has the merits of faster convergent speed and lower steady state bit error rate.

8. Extended Kalman Filter based Fuzzy Adaptive Equalizer (EKFAE)

Kalman filter is an efficient recursive filter which estimates the state of a dynamic system from a series of incomplete and noisy measurements (Kalman, R.E., 1960). Kalman filter is widely used in engineering applications from radar to computer vision and it is an important topic in control theory and control system engineering. For example, in a radar application, to track a target, information about the location, speed, and acceleration of the target is measured with a great deal of corruption by noise at any time instant. The Kalman filter exploits the dynamics of the target, which govern its time evolution, to remove the effects of the noise and obtain a good estimate of the location of the target at the past time (interpolation or smoothing process), present time (filtering process) or at future time (prediction process).

The Kalman filter is a recursive estimator. Only the estimated state from the previous time step and the current measurement are required to compute the estimate of the current state. In contrast to batch estimation techniques, no history of observation and/or estimates is required. Kalman filter has two distinct phases, which is the *predict phase* and the *update phase* (Kay, S.M., 1993). The predict phase uses the estimate from the previous time step to produce an estimate of the current state. For update phase, measurement information from

the current time step is used to refine this prediction to arrive at a new and more accurate estimate (Kay, S.M., 1993).

Kalman filter may be thought of as a sequential minimum mean square error (MMSE) estimator of a signal embedded in noise, where the signal is characterized by a dynamical or state model. If the signal and noise are jointly Gaussian, then the Kalman filter is an optimal MMSE estimator, if not, it is the optimal linear minimum mean square error (LMMSE) estimator (Kay, S.M., 1993).

The basic Kalman filter is limited to a linear assumption. However most of the real world systems are nonlinear. The nonlinearity can be associated either with the process model or with the observation model or with both. In the extended Kalman filter (EKF), the state transition and observation models need not be a linear function of the state but may instead be the differentiable functions. The functions used to compute the predicted filter state and predicted measurement cannot be applied to the covariance directly. Instead a matrix of partial derivatives, so called the Jacobian is computed. At each time step, the Jacobian is evaluated with current predicted states. This process essentially linearizes the nonlinear function around the current estimate. This result in the following extended Kalman filter equations:

Kalman Gain Matrix

$$\mathbf{G}_{\boldsymbol{\alpha}}(k) = \mathbf{E}_{\boldsymbol{\alpha}}(k-1)\boldsymbol{\Omega}_{\boldsymbol{\alpha}}{}^{T}(k)[\boldsymbol{\Omega}_{\boldsymbol{\alpha}}(k)\mathbf{E}_{\boldsymbol{\alpha}}(k-1)\boldsymbol{\Omega}_{\boldsymbol{\alpha}}{}^{T}(k)+N]^{-1} \tag{1.46}$$

(Posterior) error covariance matrix

$$\mathbf{E}_{\boldsymbol{\alpha}}(k) = \mathbf{E}_{\boldsymbol{\alpha}}(k-1) - \mathbf{G}_{\boldsymbol{\alpha}}(k)\boldsymbol{\Omega}_{\boldsymbol{\alpha}}(k)\mathbf{E}_{\boldsymbol{\alpha}}(k-1) \tag{1.47}$$

Correction

$$\boldsymbol{\alpha}(k) = \boldsymbol{\alpha}(k-1) + \mathbf{G}_{\boldsymbol{\alpha}}(k)\left[d(k) - f(\boldsymbol{x}(k))\right] \tag{1.48}$$

where k is the discrete time index;

$\boldsymbol{\alpha}(k)$ is the equalizer parameter;

$\boldsymbol{\Omega}_{\alpha}(k)$ is the Jacobian of the equalizer parameter;

$\boldsymbol{x}(k)$ is the equalizer input vector at time k;

$d(k)$ is the desired output at time k;

$f(\boldsymbol{x}(k))$ is the equalizer output at time k;

N is the measurement noise covariance.

Our extended Kalman filter based fuzzy adaptive equalizer (EKFAE) solves the following problem. Consider a real-valued scalar sequence [$d(k)$], where k =0,1,2,… is the time index, and $\boldsymbol{x}(k) \in U \equiv [C_1^-, C_1^+]\times[C_2^-, C_2^+]\times\ldots\times[C_n^-, C_n^+] \subset \Re^n$. U and $\Re^n$ are the input and output spaces of the equalizer respectively. At each time point k=0,1,2,…, we are using EKF algorithm to determine an adaptive equalizer $f_k : U \subset \Re^n \to \Re$ such that the mean square error:

$$\text{MSE} = E\{[d(k) - f_k(\boldsymbol{x}(k))]^2\} \tag{1.49}$$

is minimized.

8.1 Design procedure of the EKFAE

The EKFAE is constructed with three steps:

1. Define M fuzzy sets F_i^l for each interval $[C_i^-, C_i^+]$ of the input space, U, with Gaussian membership functions (Gupta, M.M., 2003):

$$\mu_{F_i^l} = \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right] \tag{1.50}$$

where l =1,2,…,M, i = 1,2,…,n , $x_i = x(k - I + 1)$ is the input to the equalizer, $\tilde{x}_i^l$ is the center of the i-th membership function in the l-th rule and σ_i^l represents the width of the i-th membership function in the l-th rule. $\tilde{x}_i^l$ and σ_i^l are free parameters which will be optimized using the EKF adaptation algorithm. The reason we choose Gaussian membership functions rather than triangular, trapezoidal, etc. is because, it can be shown by using Stone Weierstrass theorem, the Gaussian network is a universal approximator that can be used to uniformly approximate continuous functions on a compact set (Wang, L.X., 1992a, Gupta, M.M., 2003). However, if other types of membership functions are employed in such a network, the universal approximation capability may not be easily verified, and in the meantime, large number of rules may be needed to carry out a function approximation.

2. Construct a set of changeable fuzzy IF-THEN rules either by linguistic information or numerical information from the matching input-output data pairs:

$$R^l : \text{IF } x_1 \text{ is } F_1^l \text{ and } \ldots \text{ and } x_n \text{ is } F_n^l \text{ , THEN } d \text{ is } G^l . \tag{1.51}$$

where $d \in \Re$ is the desired output, F_i^l s′ are defined in Step 1, G^l′s are fuzzy sets defined in R^l which are determined as follows : if there are linguistic rules in the form of (1.51), set F_i^ls and G^l to be the labels of these linguistic rules, otherwise, choose μ_{G^l} and the parameters $\tilde{x}_i^l$ and σ_i^l arbitrarily. These parameters will change during the adaptation process. The rules constructed in this step are initial rules of the fuzzy adaptive equalizer. We may incorporate the linguistic rules into the EKFAE by constructing the initial equalizer base on these rules.

3. Construct the equalizer $f(x)$ base on the set of M rules by using product inference and centroid defuzzification (Gupta, M.M., 2003):

$$f(x) = \frac{\sum_{l=1}^{M} \theta^l \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right)} \tag{1.52}$$

where x =[x_1,…, x_n]^T, $\mu_{F_i^l}$s are the Gaussian membership functions defined in (1.50), and $\theta^l \in \Re$ is the value which μ_{G^l} achieves its maximum. Since we pick our

membership function, $\mu_{F_i^l}(x_i)$ to be Gaussian function which is nonzero for any $x_i \in [C_i^-, C_i^+]$, the denominator of (1.52) is nonzero for any $x \in U$. Therefore (1.52) is well defined. Since θ^l as well as $\tilde{x}_i^l$ and σ_i^l are free parameters, the equalizer (1.52) is nonlinear in the parameters.

4. The EKF algorithm as stated in (1.46), (1.47) and (1.48) is used to update the equalizer parameters $\theta^l, \tilde{x}_i^l$ and σ_i^l iteratively. This involves the computation of the Jacobian $\Omega(k)$, which is obtained as the linearization about the current value of the nonlinear parameters (θ^l, $\tilde{x}_i^l$ and σ_i^l). The Jacobians $\boldsymbol{\Omega}_\theta$, $\boldsymbol{\Omega}_{\sigma_i}$ and $\Omega_{\tilde{x}_i}$ are calculated in Appendix 1.

$$\text{Let } a^l(k-1) = \prod_{i=1}^{n} \exp\left[-\frac{1}{2}\left(\frac{x_i(k) - \tilde{x}_i^l(k-1)}{\sigma_i^l(k-1)} \right)^2 \right] \tag{1.53}$$

$$b(k-1) = \sum_{l=1}^{M} a^l(k-1) \tag{1.54}$$

$$c^l(k-1) = \frac{\left(x_i - \tilde{x}_i^l\right)^2}{\left(\sigma_i^l\right)^3} \tag{1.55}$$

$$e^l(k-1) = \frac{\left(x_i - \tilde{x}_i^l\right)}{\left(\sigma_i^l\right)^2} \tag{1.56}$$

To update parameter θ^l, we use the following EKF procedure,

$$\Omega_{\theta^l} = \frac{a^l(k-1)}{b(k-1)}, \quad \text{for } l = 1, \ldots, M \tag{1.57}$$

$$\boldsymbol{\Omega}_\theta = [\Omega_{\theta^1} \;\; \Omega_{\theta^2} \;\; \ldots \;\; \Omega_{\theta^M}]$$

$$\mathbf{G}_\theta(k) = \mathbf{E}_\theta(k-1)\boldsymbol{\Omega}_\theta^T(k)\left[\boldsymbol{\Omega}_\theta(k)\mathbf{E}_\theta(k-1)\boldsymbol{\Omega}_\theta^T(k) + N\right]^{-1} \tag{1.58}$$

$$\mathbf{E}_\theta(k) = \mathbf{E}_\theta(k-1) - \mathbf{G}_\theta(k)\boldsymbol{\Omega}_\theta(k)\mathbf{E}_\theta(k-1) \tag{1.59}$$

$$\boldsymbol{\theta}(k) = \boldsymbol{\theta}(k-1) + \mathbf{G}_\theta(k)\left[d(k) - f(\boldsymbol{x}(k))\right], \text{ where } \boldsymbol{\theta} = [\theta^1 \;\; \theta^2 \;\; \ldots \;\; \theta^M] \tag{1.60}$$

To update parameter σ_i^l, we use the following EKF procedure,

$$\Omega_{\sigma_i^l} = \frac{\left[\theta^l - f_k(x(k))\right]a^l(k-1)c^l(k-1)}{b(k-1)}, \quad \text{for } l = 1, \ldots, M \tag{1.61}$$

$$\boldsymbol{\Omega}_{\sigma_i} = [\Omega_{\sigma_i^1} \ \Omega_{\sigma_i^2} \ \ldots \ \Omega_{\sigma_i^M}]$$

$$\mathbf{G}_{\sigma_i}(k) = \mathbf{E}_{\sigma_i}(k-1)\boldsymbol{\Omega}_{\sigma_i}{}^T(k)\left[\boldsymbol{\Omega}_{\sigma_i}(k)\mathbf{E}_{\sigma_i}(k-1)\boldsymbol{\Omega}_{\sigma_i}{}^T(k) + N\right]^{-1} \tag{1.62}$$

$$\mathbf{E}_{\sigma_i}(k) = \mathbf{E}_{\sigma_i}(k-1) - \mathbf{G}_{\sigma_i}(k)\boldsymbol{\Omega}_{\sigma_i}(k)\mathbf{E}_{\sigma_i}(k-1) \tag{1.63}$$

$$\boldsymbol{\sigma}_i(k) = \boldsymbol{\sigma}_i(k-1) + \mathbf{G}_{\sigma_i}(k)\left[d(k) - f(\boldsymbol{x}(k))\right], \text{ where } \boldsymbol{\sigma}_i = [\sigma_i^{\,1} \ \sigma_i^{\,2} \ \ldots \ \sigma_i^{\,M}] \tag{1.64}$$

To update parameter $\tilde{x}_i^{\,l}$, we use the following EKF procedure,

$$\Omega_{\tilde{x}_i^l} = \frac{\left[\theta^l - f_k(x(k))\right]a^l(k-1)e^l(k-1)}{b(k-1)}, \text{ for } l = 1,\ldots, M \tag{1.65}$$

$$\boldsymbol{\Omega}_{\tilde{x}_i} = [\Omega_{\tilde{x}_i^1} \ \Omega_{\tilde{x}_i^2} \ \ldots \ \Omega_{\tilde{x}_i^M}]$$

$$\mathbf{G}_{\tilde{x}_i}(k) = \mathbf{E}_{\tilde{x}_i}(k-1)\boldsymbol{\Omega}_{\tilde{x}_i}{}^T(k)\left[\boldsymbol{\Omega}_{\tilde{x}_i}(k)\mathbf{E}_{\tilde{x}_i}(k-1)\boldsymbol{\Omega}_{\tilde{x}_i}{}^T(k) + N\right]^{-1} \tag{1.66}$$

$$\mathbf{E}_{\tilde{x}_i}(k) = \mathbf{E}_{\tilde{x}_i}(k-1) - \mathbf{G}_{\tilde{x}_i}(k)\boldsymbol{\Omega}_{\tilde{x}_i}(k)\mathbf{E}_{\tilde{x}_i}(k-1) \tag{1.67}$$

$$\tilde{\boldsymbol{x}}_i(k) = \tilde{\boldsymbol{x}}_i(k-1) + \mathbf{G}_{\tilde{x}_i}(k)\left[d(k) - f(\boldsymbol{x}(k))\right], \text{ where } \tilde{\boldsymbol{x}}_i = [\tilde{x}_i^{\,1} \ \tilde{x}_i^{\,2} \ \ldots \ \tilde{x}_i^{\,M}] \tag{1.68}$$

The EKFAE algorithm is summarized in Table 1.

Initialize The number of iteration, n. $\boldsymbol{\theta} = [\theta^1\ \theta^2 \ldots \theta^M]$ in the range of [-0.5,0.5], $\boldsymbol{\sigma}_i = [\sigma_i^{\,1} \ \sigma_i^{\,2} \ \ldots \ \sigma_i^{\,M}]$ in the range of [0.1,0.3], $\tilde{\boldsymbol{x}}_i = [\tilde{x}_i^{\,1} \ \tilde{x}_i^{\,2} \ \ldots \ \tilde{x}_i^{\,M}]$ in the range of [-2.0,2.0], and $\mathbf{E}_\theta(0)$, $\mathbf{E}_{\sigma_i}(0)$ and $\mathbf{E}_{\tilde{x}_i}(0)$ each equals to an identity matrix of $M \times M$ elements. For k=1,2,…,*n*, do 1. Calculate μ_{F^l} using (1.50), and $a^l(k\text{-}1)$, $b^l(k\text{-}1)$, $c^l(k\text{-}1)$ and $e^l(k\text{-}1)$ using (1.53)-(1.56). 2. Calculate the Jacobians $\Omega_\theta(k)$, $\Omega_{\sigma_i}(k)$ and $\Omega_{x_i}(k)$ using (1.57), (1.61) and (1.65) respectively. 3. Calculate the Kalman gain matrices $\mathbf{G}_\theta(k)$, $\mathbf{G}_{\sigma_i}(k)$ and $\mathbf{G}_{\tilde{x}_i}(k)$ using (1.58), (1.62) and (1.66) respectively. 4. Update the error covariance matrices $\mathbf{E}_\theta(k)$, $\mathbf{E}_{\sigma_i}(k)$ and $\mathbf{E}_{\tilde{x}_i}(k)$ using (1.59), (1.63) and (1.67) respectively. 5. Update the parameters, $\boldsymbol{\theta}(k)$, $\boldsymbol{\sigma_i}(k)$ and $\tilde{\boldsymbol{x}}_i(k)$, using (1.60), (1.64) and (1.68) respectively.

Table 1. The EKFAE algorithm

9. Performance evaluation

In this section, we investigate the performance EKFAE and three other types of short listed adaptive equalizers, namely: decision feedback recursive least-squares adaptive equalizer, recursive least squares (RLS) based FAE and least mean squares (LMS) based FAE in terms of convergence speed and steady-state bit error rate (BER). We consider an arbitrary communication channel with transfer function $h(z) = 1 + 0.5z^{-1} + 0.3z^{-2} - 0.1z^{-3}$. Spread spectrum modulation/demodulation technique is use with spreading sequence [1 1 –1 1]. The DF-RLS adaptive equalizer has a number of forward taps N_f = 2 and number of feedback taps N_b = 2, while the order and lag of the FAEs are n=2 and d=0 respectively. The background noise is modeled as zero mean Gaussian with variance σ^2. As shown in Section 1.11, a smaller number of rules can be used to reduce the computational complexity of the FAEs. To ensure a fair comparison of the different FAEs, we have considered M = 25 rules for the EKFAE, the RLS-based FAE ($m_1 = m_2 = 5$) and the LMS-based FAE. We have also compared the performance of those equalizers with the optimum MAP equalizer that achieves the minimum probability of error.

Example 1. We first consider the situation where there is no linguistic information for the FAEs. We randomly set $\theta^l(0)$ in [-0.5 0.5], $\tilde{x}_i^l(0)$ in [-2.0 2.0] and $\sigma_i^l(0)$ in [0.1 0.3]. For the EKFAE, we set N = 0.999 and $E(0) = I$, where I is an identity matrix of $M \times M$. For the LMS-based FAE, we set the step size for adaptation, α = 0.05. For DF-RLS equalizer and the RLS-based FAE, we set the forgetting factor, λ = 0.999 and σ = 0.01. When signal to noise ratio equals to 10dB, we plot the convergence curves for these equalizers in Fig. 12. Around 30 times of experiments were simulated in order to obtain the convergence curves. The simulation results show that without using any linguistic information, the EKFAE is a well performing nonlinear adaptive equalizer. The EKFAE achieves the fastest convergent speed. The EKFAE also outperforms the DF-RLS adaptive equalizer, RLS-based FAE and LMS-based FAE in term of steady-state BER.

Example 2. Next, we incorporate some linguistic information about the decision region into the fuzzy adaptive equalizers. For EKFAE and LMS-based FAE, the first M = 25 training data pairs are used to construct the initial fuzzy rules. In particular, the centers of the fuzzy antecedents $\tilde{x}^l = [\tilde{x}_1^l \ \tilde{x}_2^l]$ are set to the received signals $[x(l) \ x(l-1)]$ where l = 1,2,…,M. Similarly, the centers of the fuzzy consequences θ^l are set to the corresponding scaled desired data, $\beta d(l)$, where l = 1,2,…,M. In the simulation, we use $\beta = 0.5$. For RLS-based FAE, we use the same 25 training data pairs to construct the initial fuzzy rules. The centers for the fuzzy antecedents, $\tilde{x}_i^{ji}$ =-1.6, -0.8, 0, 0.8, 1.6 are fixed and will not change during the adaptation, where ji=1,2,…,5 and i=1,2. The decision region of the RLS-based FAE can be partitioned as shown in Fig. 11. Every section corresponds to a certain fuzzy rule. The initial fuzzy rules can be constructed as follows: if the equalizer inputs $[x(l) \ x(l-1)]$ fall in the region that corresponds to the rule $R^{(j1,j2)}$, then set the center of the fuzzy consequence $\theta^{(j1,j2)}$ to the corresponding scaled desired data, $\beta d(l)$. For the region that the 25 received signal does not fall in, we choose $\theta^{(j1,j2)}$ arbitrarily in [-0.3 0.3]. Other parameters remain the same as in *Example 1.* When signal to noise ratio equals to 10 dB, we plot the convergence curves for the three equalizers in Fig. 13. Around 30 times of experiments were

simulated in order to obtain the convergence curves. The simulation results show that by incorporating some noisy linguistic information about the decision region into the equalizers, the convergence rates of the three fuzzy adaptive equalizers are greatly improved. We observed that EKFAE achieves the fastest convergence and lowest steady-state BER compared with the LMS-based FAE and RLS-based FAE.

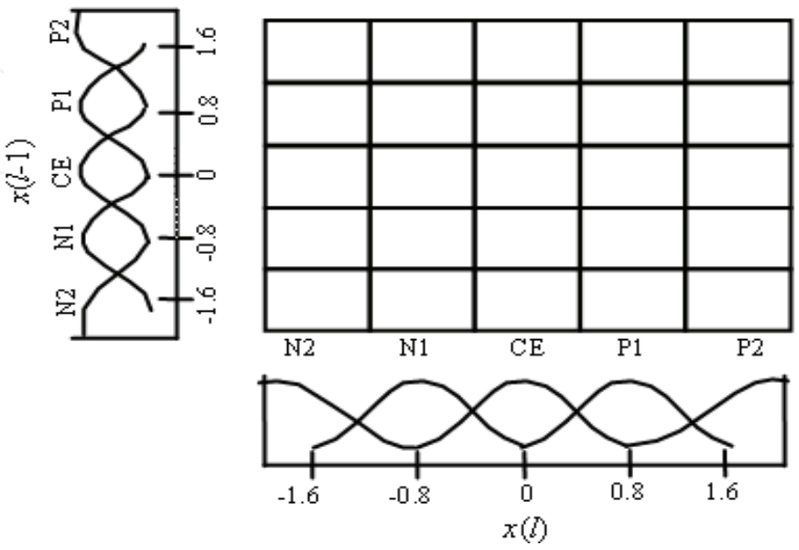

Fig. 11. Construction of fuzzy rules for the fuzzy RLS-based FAE.

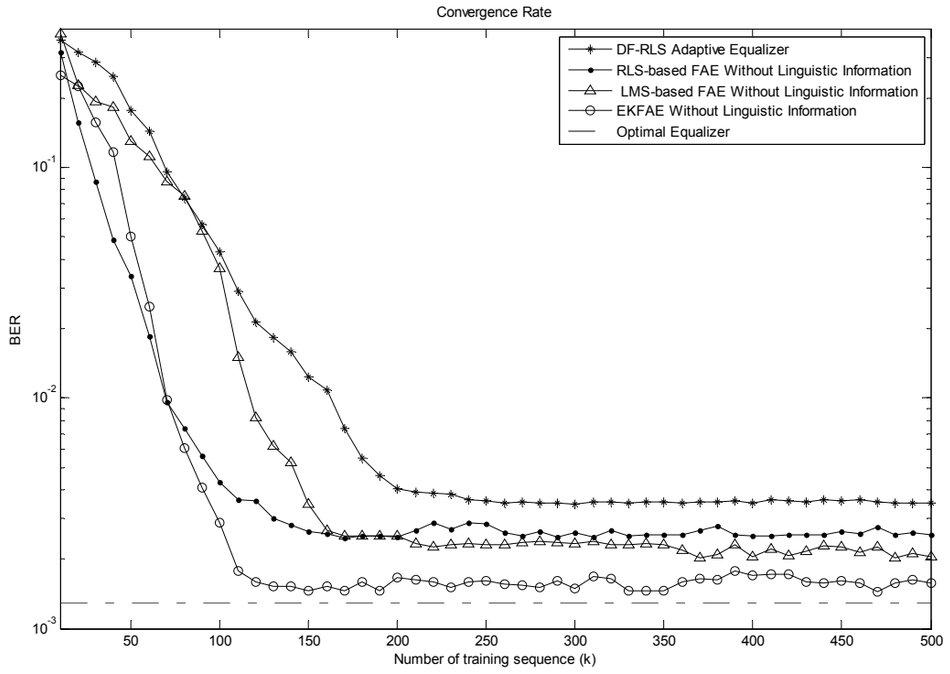

Fig. 12. Convergence rate for DF-RLS adaptive equalizers and 3 types of fuzzy adaptive equalizers without linguistic information.

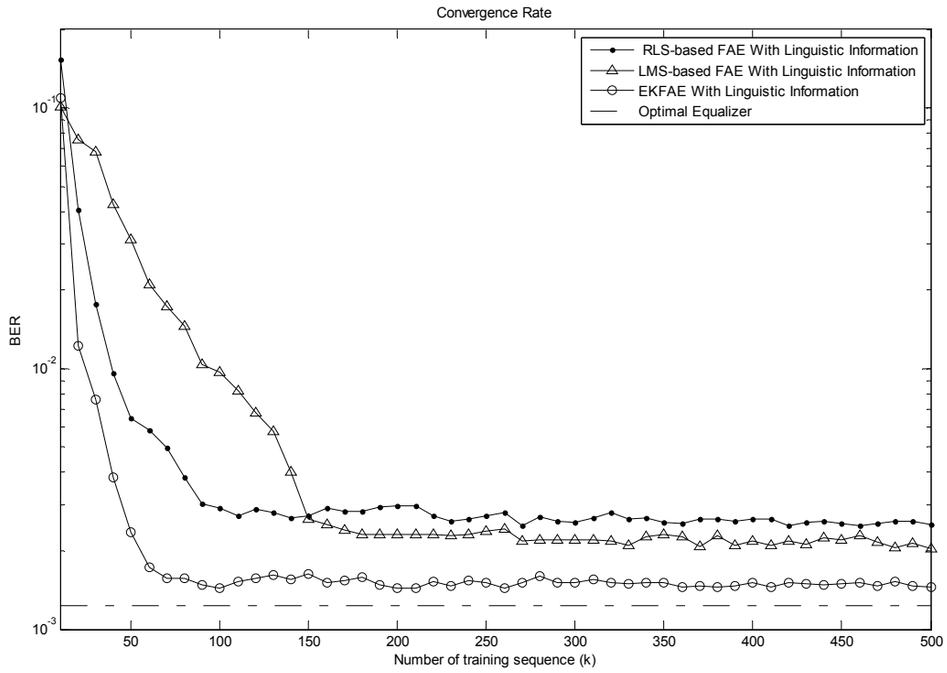

Fig. 13. Convergence rate for 3 types of fuzzy adaptive equalizers with linguistic information.

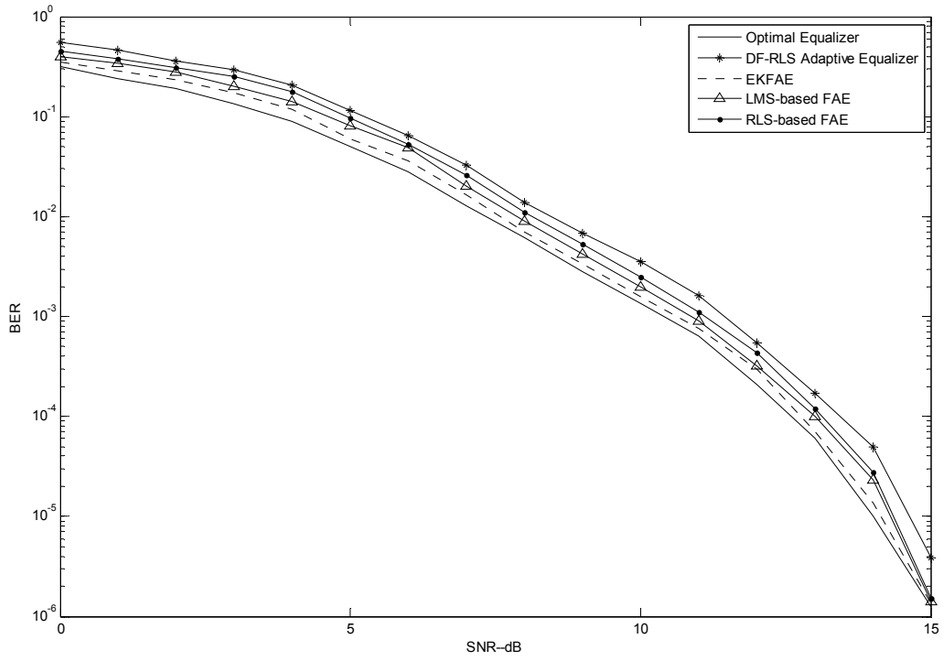

Fig. 14. The bit error rate vs. signal to noise ratio plots of DF-RLS equalizer, EKFAE, RLS-based FAE, LMS-based FAE, and the optimal MAP equalizer

Example 3. Finally, the bit error rates of DF-RLS adaptive equalizer, EKFAE, LMS-based FAE and RLS-based FAE are compared with the maximum-a-posteriori probability (MAP) equalizer that has the minimum probability of error. Fig. 14 shows the bit error rates of the optimum MAP equalizer and the EKFAE for different signal to noise ratios. We see that the bit error rate of the EKFAE is the closest to the optimal one.

10. Computational complexity

In this section, we study the computational complexity of the DF-RLS adaptive equalizer and the three FAEs (i.e., RLS, LMS and EKF based FAEs). During the training process of DF-RLS adaptive equalizer, LMS-based FAE, RLS-based FAE and EKFAE, the computational requirement for one training iteration is summarized and listed in Table 2..

The computational requirements for the adaptive equalizers are dependent on the order of the equalizers (N_f and N_b for DF-RLS adaptive equalizer and n for FAEs) and the number of rules ($m^n = m_1.m_2.....m_n$ for RLS-based FAE and M for LMS-based FAE and EKFAE). If we set the adaptive equalizer with the same order ($N_f+N_b = n$), then $N_f+N_b << m^n$ and M. Hence from Table 2, we can see that DF-RLS adaptive equalizer has the least time complexity among the adaptive equalizers. The computational complexity of the LMS based FAE is on the order of O(nM) while EKFAE is on the order of O(nM^2) during training. EKFAE has higher time complexity compared to LMS based FAE. RLS based FAE is on the order of O(m^{2n}). If we set $m^n = M$, then RLS based FAE is on the order of (O(M^2)). Among the three FAEs, EKFAE has the high computational requirement for training process (O(nM^2)), followed by RLS based FAE (O(M^2)),whereas LMS-based FAE has the lowest computational requirement for training process (O(nM)) .

After DF-RLS adaptive equalizer, LMS-based FAE, RLS-based FAE and EKFAE complete their training, their parameters are fixed and the actual estimation for desired symbol goes on. The computational complexity per estimated symbol for the above equalizers is listed in Table 3. From Table 3, DF-RLS adaptive equalizer has the least time complexity if $N_f + N_b = n$. We can observe that both LMS based FAE and EKFAE are in the same time complexity category (O(nM)) in symbol estimation after the training process stopped. Since $m^n = M$, RLS based FAE with O(nm^n) is in the same time complexity category with EKFAE (O(nM)) for symbol estimation. We can conclude that the DF-RLS adaptive has the least time complexity, whereas the three FAEs are in the same time complexity category involved for estimation of one desired symbol if we set $N_f + N_b = n$, $m=m_1=m_2=\ldots=m_n >1$ and $m^n = M$.

Equalizer Type	Addition/Subtraction	Multiplication	Division	e^{-x}
DF-RLS adaptive equalizer	$4(N_f+N_b)^2+(N_f+N_b)$	$5(N_f+N_b)^2+6(N_f+N_b)-1$	2	0
RLS-based FAE	$4m^{2n}+(1+n)m^n-1$	$5m^{2n}+(6+3n)m^n$	$(1+n)m^n+2$	nm^n
LMS-based FAE	$(9n+4)M-2$	$(16n+2)M$	$(5n+1)M+1$	nM
EKFAE	$(8n+4)M^2+(3n+2)M-2$	$(8n+4)M^2+(21n+3)M$	$(5n+1)M+2n+2$	nM

Table 2. Computational requirements for DF-RLS adaptive equalizer, RLS-based FAE and EKFAE for 1 training iteration.

Equalizer Type	Addition/Subtraction	Multiplication	Division	e^{-x}
DF-RLS adaptive equalizer	N_f+N_b-1	N_f+N_b	0	0
RLS based FAE	$(n+2)m^n$ -2	$(3n+3)m^n$	$(n+1)m^n$	nm^n
LMS-based FAE	$(n+2)M$-2	$4nM$	$nM+1$	nM
EKFAE	$(n+2)M$-2	$4nM$	$nM+1$	nM

Table 3. Computational requirements for DF-RLS adaptive equalizer, RLS-based FAE and EKFAE per desired symbol estimation

11. Number of rules

In this section, we investigate the number of rules required for the FAEs. The design parameters of the proposed equalizer are the number of fuzzy IF-THEN rules, M, and the initial values of the parameters $\tilde{\mathbf{x}}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\theta}$. If the channel impulse response is known at the receiver, the exact number of rules, M can be calculated as:

$$M = 2^{L+n} \tag{1.69}$$

where L is the ISI span (in bits) and n denotes the order of the equalizer. The ISI span, L, depends on the length of the channel impulse response, N, and the length of the spreading sequence, N_s. For $N = 1$, $L = 1$; for $1 < N \le N_s + 1$, $L = 2$; for $N_s + 1 < N \le 2N_s + 1$, $L = 3$; and so on. We shall look at one example.

Consider a channel with impulse response $h(z) = 1 + 0.5z^{-1} + 0.3z^{-2} - 0.1z^{-3}$. The spreading sequence is [1 1 –1 1]. The order and lag of the EKFAE are $n = 2$ and $d = 0$, respectively. In this case, the number of noiseless equalizer input signal states is $M = 2^{2+2} = 16$. Fig. 15a shows the noiseless signal states. When noise is added, the equalizer inputs form 16 clusters with centers equal to the noiseless states. The noisy signal states are illustrated in Fig. 15b. Since there are 16 clusters and one fuzzy rule is required to cater for one cluster, a total of 16 fuzzy rules are needed. It is shown in (Patra, S.K., 1998) that the FAE is optimal when the number of rules is equal to the number of noiseless states, and the parameters, $\tilde{\mathbf{x}}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\theta}$, are set to the locations of the noiseless states, the standard deviation of the noise, and the corresponding desired data symbols of the noiseless states, respectively.

However, in practice, the channel impulse response is unknown at the receiver. To determine a suitable number of rules for the EKFAE, the channel impulse response has to be estimated first. Next, by using the same channel as above, we study the effects of the channel estimation on the performance of the EKFAE. The BER performances of the EKFAE in different channel conditions are plotted against the estimated initial rules in Fig. 16. When the number of rules is 16 the equalizer achieves the optimum performance. Also, when the number of rules is more than 10, the BER performance is very near to optimum. The general conclusion is, the number of rules need not be equal to the optimum number of rules exactly. Consequently, the EKFAE is robust against the channel estimation error. From our experience, using 70% of the optimum number of rules is often sufficient to give good results.

When the channel is unknown, the initial values of the parameters $\tilde{\mathbf{x}}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\theta}$, are set randomly in their valid range. Then, these parameters are optimized by the EKF adaptation

algorithm in the MMSE sense. From extensive experiments, the EKFAE never failed to converge with random initial values. However, faster convergence can be achieved if some knowledge about the channel is incorporated into the setting of the initial parameter values.

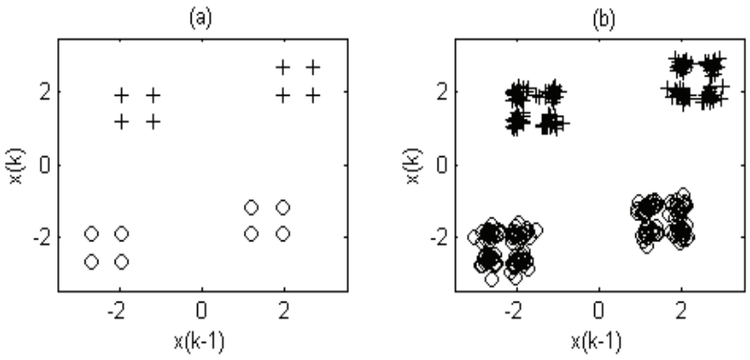

Fig. 15. Equalizer input signal states. The "+" and "o" denote the signals corresponding to "+1" and "-1" desired data bits, respectively. (a) Noiseless states. (b) Noisy states.

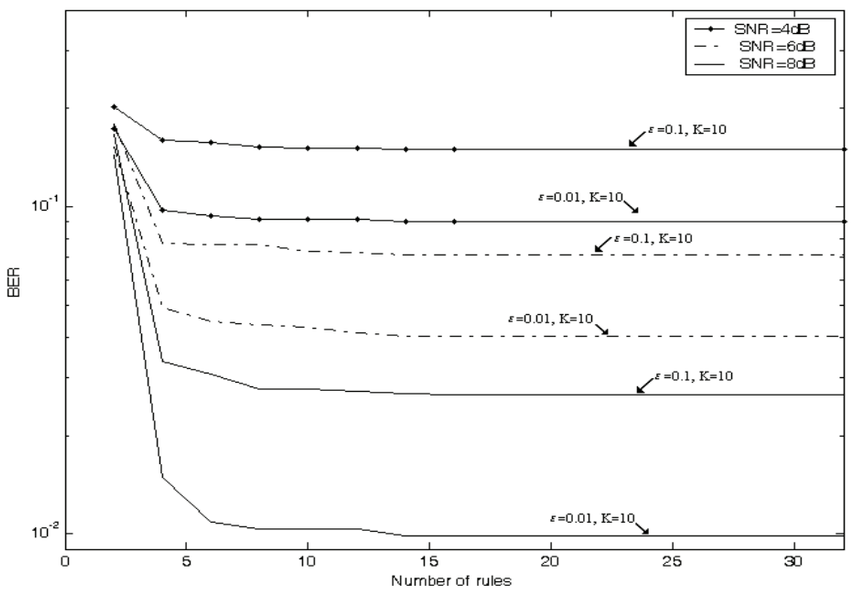

Fig. 16. The BER performances of the EKFAE in different SNRs versus number of initial rules

12. Summary

Fundamentals of fuzzy logic, fuzzy logic system, equalizer, adaptive equalizer and fuzzy adaptive equalizer are described in this chapter. A new fuzzy adaptive equalizer with extended Kalman filter adaptation (EKFAE) had been derived in this chapter for communication channel equalization. The merits of EKFAE as compared with three other types of short listed adaptive equalizers, namely: decision feedback recursive least-squares adaptive equalizer, recursive least squares (RLS) based FAE and least mean squares (LMS) based FAE are faster convergent speed and lower steady state BER. Simulation results in

Section 1.9 show that EKFAE converges fastest without using any linguistic information. By incorporating some linguistic rules about the equalizer inputs into EKFAE, the adaptation speed is greatly improved. The steady state BER of EKFAE is very close to that of the optimal equalizer. The computational requirements for those adaptive equalizers in consideration (DF-RLS adaptive equalizer, RLS, LMS and EKF based FAEs) are calculated out. The number of rules required for the FAEs are also investigated.

13. References

Al-Holou, N. T., Lahdhiri, Joo, D., Weaver, J., Al-Abbas, F. (2002). Sliding mode neural network inference fuzzy logic control for active suspension systems. *IEEE Trans. on Fuzzy Syst.*.p.p.234-247.

Bhatti, S. S. (2002). What is neuro fuzzy-logic*? The Tribune (Chandigarh, India).*

Chen, S., Gibson, G.J., Cowan, C.F.N., Grant, P.M. (1990). Adaptive Equalization of Finite Nonlinear Channels Using Multilayer Perceptrons. *Signal Processing, Vol. 20*, p.p.107-119.

Chen, S., Gibson, G. J., Cowan, C. F. N., Grant, P. M. (1991). Reconstruction of binary signals using an adaptive radial-basis-function equalizer. *Signal Processing, Vol. 22*. p.p.77-93.

Cowan, C. F. N., Grant, P. M. (1985). Adaptive Filters. *Englewood Cliffs, NJ: Prentice Hall*.

Eichfeld, H., Kunermund, T., Menke, M. (1996). A 12b general-purpose fuzzy logic controller chip. *IEEE Trans. on Fuzzy Syst., Vol. 4*. p.p.460-475.

Gupta, M. M., Qi, J. (1991). Theory of T-norms and fuzzy inference methods. *Fuzzy sets and systems, Vol. 40*. p.p.431-450.

Gupta, M. M., Rao, D. H. (1994[a]). Neuro-Control Systems: Theory and Applications. *IEEE Press, NJ.*

Gupta, M. M., Rao, D. H. (1994[b]). On the principles of fuzzy neural networks. *Fuzzy sets and systems, Vol. 61*. p.p.1-18.

Gupta, M. M., Jin, L., Homma, N. (2003). Static And Dynamic Neural Networks, From Fundamentals to Advanced Theory. *IEEE Press, John Wiley & Sons Inc*.

Haykin, S. (2002). Adaptive Filter Theory. *4th Ed. Prentice Hall*.

Jang, J. S. R., Sun, C. T. (1993). Functional equivalence between radial basis-function networks and fuzzy inference systems. *IEEE Trans. on Neural Networks 4(1).* p.p.156-159.

Kallman, H. J. (1940). Transversal filters*. Proc. IRE, Vol. 28*. p.p.302-310.

Kalman, R. E. (1960). A new Approach to Linear Filtering and Prediction Problems. *Trans. of the ASME, Journal of Basic Engineering, Vol. 82*. p.p.35-45.

Kaufmann, A., Gupta, M. M. (1988). Fuzzy Mathematical Models in Engineering and Management Science. *Elsevier, Amsterdan.*

Kay, S. M. (1993). Fundamentals Of Statistical Signal Processing: Estimation Theory. *Prentice Hall Signal Processing Series, Vol. 1*. p.p.419-477.

Kong, X., Han, C., Wei, R. (2004). A Stable Total Least Square Adaptive Algorithm for the Nonlinear Volterra Filter. *The 7th International Conference on Information Fusion, Stockholm, Sweden, Jun 28- July 1,2004*. p.p.518-522.

Mastorakis, N. E. (2004). Modeling Dynamical Systems via the Takagi-Sugeno Fuzzy Model. *Procs. of the 4th WSEAS Int. Conf. on Fuzzy sets and Fuzzy Systems, Udine, Italy*.

Mathews, V. J. (1991). Adaptive polynomial filters. *IEEE Signal Processing Magazine, Vol.8, No.3*, p.p.10-26.

Meisel, W. S. (1969). Potential functions in mathematical pattern recognition. *IEEE Trans. Comput., Vol. 18, No. 10*. p.p.911-918.

Mendel, J. M. (1995). Fuzzy Logic Systems for Engineering: A Tutorial. *Proceedings of the IEEE, Vol. 83, No. 3*. p.p.345-377.

Mendel , J. M., Mouzouris, G. C. (1997). Designing Fuzzy Logic Systems. *IEEE Trans. on Circuits and Systems II : Analog and Digital Signal Processing, Vol.44, No.11*.

Patra, S. K., Mulgrew B. (1998). Efficient architecture for Bayesian Equalization Using Fuzzy Filters. *IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 45*. p.p.812-820.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: A review. *Algorithms for Approximation, J. C. Mason and M. G. Cox, Eds. Oxford: Oxford University Press*. p.p.143-167.

Proakis, J. G. (1995). Digital Communications. *3rd ed. New York: McGraw-Hill*.

Sailer, T.M. (2001). Decision Feedback Equalization for Powerline and HIPERLAN. *PhD thesis, Swiss Federal Institute of Technology Zurich. Diss. ETH No. 13970*.

Seung, S. (2002). Multilayer perceptrons and backpropagation learning. *9.641 Lecture4*. 1-6. Source from: http://hebb.mit.edu/courses/9.641/2002/lectures/lecture04.pdf

Tanaka, K., Sugeno, M. (1998). Introduction to fuzzy modeling. *Fuzzy Syst. Modeling and Control, H. T. Nguyen and M. Sugeno, Eds. Boston, MA: Kluwer*. p.p.63-89.

Wang, L. X. (1992[a]). Fuzzy systems are universal approximators. *Proc. IEEE Conf. Fuzzy Systems (San Diego)*. p.p.1163-1170.

Wang, L. X., Mendel, J. M. (1992[b]). Fuzzy Basis Function, Universal Approximation, and Orthogonal Least Squares Learning. *IEEE Trans. on Neural Networks, Vol. 3, No. 5*. p.p.807-814.

Wang, L. X., Mendel, J. M. (1993). Fuzzy Adaptive Filters, with Application to Nonlinear Channel Equalization. *IEEE Trans. On Fuzzy Systems, Vol. 1, No. 3*. p.p.161-170.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control 8*. p.p.338-353.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences, Vol. 8*. p.p.199-249.

Appendix 1:

Derivation of Extended Kalman Filter based adaptation algorithm for fuzzy adaptive equalizer.

Let $$a^l(k-1)=\prod_{i=1}^{\eta}\exp\left[-\frac{1}{2}\left(\frac{x_i(k)-\tilde{x}_i^{\,l}(k-1)}{\sigma_i^{\,l}(k-1)}\right)^2\right] \tag{A1.1}$$

$$b(k-1)=\sum_{l=1}^{M}a^l(k-1) \tag{A1.2}$$

$$c^l(k-1)=\frac{\left(x_i-\tilde{x}_i^{\,l}\right)^2}{\left(\sigma_i^{\,l}\right)^3} \tag{A1.3}$$

$$e^l(k-1)=\frac{\left(x_i-\tilde{x}_i^{\,l}\right)}{\left(\sigma_i^{\,l}\right)^2} \tag{A1.4}$$

Therefore, the equalizer output as from equation (1.52) becomes:

$$f_k(x(k))=\frac{\sum_{l=1}^{M}\theta^l\left(\prod_{i=1}^{\eta}\mu_{F_i^l}(x_i)\right)}{\sum_{l=1}^{M}\left(\prod_{i=1}^{\eta}\mu_{F_i^l}(x_i)\right)}=\frac{\sum_{l=1}^{M}\theta^l\left(\prod_{i=1}^{\eta}\exp\left[-\frac{1}{2}\left(\frac{x_i-\tilde{x}_i^{\,l}}{\sigma_i^{\,l}}\right)^2\right]\right)}{\sum_{l=1}^{M}\left(\prod_{i=1}^{\eta}\exp\left[-\frac{1}{2}\left(\frac{x_i-\tilde{x}_i^{\,l}}{\sigma_i^{\,l}}\right)^2\right]\right)}=\frac{\sum_{l=1}^{M}\theta^l a^l(k-1)}{b(k-1)} \tag{A1.5}$$

The computation of the Jacobian $\Omega(k)$ requires the evolution of the partial derivatives of $f_k(x(k))$ vs. the parameters of the fuzzy network:

$$\Omega_{\theta^l} = \frac{\partial f_k(x(k))}{\partial \theta^l} = \frac{\partial}{\partial \theta^l}\left(\frac{\sum_{l=1}^{M}\theta^l a^l(k-1)}{b(k-1)}\right) = \frac{a^l(k-1)}{b(k-1)} \tag{A1.6}$$

$$\Omega_{\sigma_i^l} = \frac{\partial f_k(x(k))}{\partial \sigma_i^{\,l}} = \frac{\partial f_k(x(k))}{\partial \mu_{F_i^l}} \times \frac{\partial \mu_{F_i^l}}{\partial \sigma_i^{\,l}}$$

since $$\frac{\partial f_k(x(k))}{\partial \mu_{F_i^l}} = \frac{\partial}{\partial \mu_{F_i^l}}\left(\frac{\sum_{l=1}^{M}\theta^l\prod_{j=1}^{\eta}\mu_{F_j^{\,l}}}{\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^{\,l}}}\right) = \frac{\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}\theta^l\prod_{\substack{j=1\\ j\neq i}}^{\eta}\mu_{F_j^l} - \sum_{l=1}^{M}\theta^l\prod_{j=1}^{\eta}\mu_{F_j^l}\prod_{\substack{j=1\\ j\neq i}}^{\eta}\mu_{F_j^l}}{\left(\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}\right)^2}$$

$$= \frac{\prod_{\substack{j=1\\ j\neq i}}^{\eta}\mu_{F_j^l}\left[\theta^l\left(\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}\right) - \sum_{l=1}^{M}\theta^l\prod_{j=1}^{\eta}\mu_{F_j^l}\right]}{\left(\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}\right)^2} = \frac{\prod_{\substack{j=1\\ j\neq i}}^{\eta}\mu_{F_j^l}}{\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}}\left[\theta^l - f_k(x(k))\right]$$

and $$\frac{\partial \mu_{F_i^l}}{\partial \sigma_i^l} = \frac{\partial}{\partial \sigma_i^l}\left(\exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^{\,l}}{\sigma_i^l}\right)^2\right]\right)$$

$$\frac{\partial \mu_{F_i^l}}{\partial \sigma_i^l} = \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^{\,l}}{\sigma_i^{\,l}}\right)^2\right] \times \frac{\partial}{\partial \sigma_i^{\,l}}\left(-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^{\,l}}{\sigma_i^{\,l}}\right)^2\right)$$

$$= \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^{\,l}}{\sigma_i^{\,l}}\right)^2\right] \times \left(-\frac{1}{2}\right) \times \left(x_i - \tilde{x}_i^{\,l}\right)^2 \times \frac{\partial}{\partial \sigma_i^{\,l}}\left(\frac{1}{\left(\sigma_i^{\,l}\right)^2}\right)$$

$$= \frac{\left(x_i - \tilde{x}_i^{\,l}\right)^2}{\left(\sigma_i^{\,l}\right)^3}\exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^{\,l}}{\sigma_i^{\,l}}\right)^2\right] = \mu_{F_i^l}\frac{\left(x_i - \tilde{x}_i^{\,l}\right)^2}{\left(\sigma_i^l\right)^3}$$

Therefore,

$$\Omega_{\sigma_i^l} = \frac{\prod_{\substack{j=1\\ j\neq i}}^{\eta}\mu_{F_j^l}}{\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}} \times \left[\theta^l - f_k(x(k))\right] \times \mu_{F_j^l} \times \frac{\left(x_i - \tilde{x}_i^{\,l}\right)^2}{\left(\sigma_i^l\right)^3} = \frac{\left[\theta^l - f_k(x(k))\right]\prod_{j=1}^{\eta}\mu_{F_j^l}\frac{\left(x_i - \tilde{x}_i^{\,l}\right)^2}{\left(\sigma_i^l\right)^3}}{\sum_{l=1}^{M}\prod_{j=1}^{\eta}\mu_{F_j^l}}$$

$$= \frac{\left[\theta^l - f_k(x(k))\right] a^l(k-1) c^l(k-1)}{b(k-1)} \tag{A1.7}$$

$$\Omega_{\tilde{x}_i^l} = \frac{\partial f_k(x(k))}{\partial \tilde{x}_i^l} = \frac{\partial f_k(x(k))}{\partial \mu_{F_i^l}} \times \frac{\partial \mu_{F_i^l}}{\partial \tilde{x}_i^l}$$

$$\text{since } \frac{\partial f_k(x(k))}{\partial \mu_{F_i^l}} = \frac{\prod_{\substack{j \neq i \\ j=1}}^{\eta} \mu_{F_j^l}}{\sum_{l=1}^{M} \mu_{F_j^l}} \left[\theta^l - f_k(x(k))\right]$$

$$\text{and } \frac{\partial}{\partial \tilde{x}_i^l}\left(\exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right]\right) = \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right] \times \frac{\partial}{\partial \tilde{x}_i^l}\left(-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right)$$

$$= \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right] \times \left(-\frac{1}{2\left(\sigma_i^l\right)^2}\right) \times \frac{\partial}{\partial \tilde{x}_i^l}\left(x_i - \tilde{x}_i^l\right)^2$$

$$= \exp\left[-\frac{1}{2}\left(\frac{x_i - \tilde{x}_i^l}{\sigma_i^l}\right)^2\right] \frac{\left(x_i - \tilde{x}_i^l\right)}{\left(\sigma_i^l\right)^2}$$

$$= \mu_{F_i^l} \frac{\left(x_i - x_i^l\right)}{\left(\sigma_i^l\right)^2}$$

Therefore,

$$\Omega_{\tilde{x}_i^l} = \frac{\prod_{\substack{j \neq i \\ j=1}}^{\eta} \mu_{F_j^l}}{\sum_{l=1}^{M} \prod_{j=1}^{\eta} \mu_{F_j^l}} \times \left[\theta^l - f_k(x(k))\right] \times \mu_{F_j^l} \times \frac{\left(x_i - \tilde{x}_i^l\right)}{\left(\sigma_i^l\right)^2}$$

$$= \frac{\left[\theta^l - f_k(x(k))\right] \prod_{j=1}^{\eta} \mu_{F_j^l} \frac{\left(x_i - \tilde{x}_i^l\right)}{\left(\sigma_i^l\right)^2}}{\sum_{l=1}^{M} \prod_{j=1}^{\eta} \mu_{F_j^l}}$$

$$= \frac{\left[\theta^l - f_k(x(k))\right] a^l(k-1) e^l(k-1)}{b(k-1)} \tag{A1.8}$$

Kalman Filter Recent Advances and Applications

Edited by Victor M. Moreno and Alberto Pigazo

ISBN 978-953-307-000-1
Hard cover, 584 pages
Publisher InTech
Published online 01, April, 2009
Published in print edition April, 2009

The aim of this book is to provide an overview of recent developments in Kalman filter theory and their applications in engineering and scientific fields. The book is divided into 24 chapters and organized in five blocks corresponding to recent advances in Kalman filtering theory, applications in medical and biological sciences, tracking and positioning systems, electrical engineering and, finally, industrial processes and communication networks.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Wai Kit Wong and Heng Siong Lim (2009). Extended Kalman Filter Based Fuzzy Adaptive Filter, Kalman Filter Recent Advances and Applications, Victor M. Moreno and Alberto Pigazo (Ed.), ISBN: 978-953-307-000-1, InTech, Available from: http://www.intechopen.com/books/kalman_filter_recent_adavnces_and_applications/extended_kalman_filter_based_fuzzy_adaptive_filter

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen